Dual-Domain Image Denoising

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Claude Knaus

von Hemberg

Leiter der Arbeit:
Prof. Dr. M. Zwicker
Institut für Informatik und angewandte Mathematik

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern,

Der Dekan:
Prof. Dr. S. Decurtins

to my parents

Abstract

Image denoising is a classic problem in digital image processing, which is half a century old. Despite its age, it remains an active subject. State-of-the-art image denoising methods produce objectively excellent results and approach theoretical limits. However, they still suffer from visually disturbing artifacts. While they produce acceptable results for natural images, human eyes are less forgiving when viewing synthetic images. Their success in denoising natural images is believed to be based on the use of patches, and the ability to learn statistics about them. Unfortunately, patch-based algorithms producing high-quality results are sophisticated and difficult to implement.

We present a class of new image denoising algorithms that does not use patches and yet produces high-quality images. Our algorithms produce competitive results for grayscale images and surpass the state-of-the-art for color images. Unlike other state-of-the-art methods, the methods presented in this thesis produce nearly artifact free images, making them suitable for denoising synthetic images. Our methods are based on a new filter which operates in the spatial domain and in the frequency domain. This new filter is versatile and can also be used for removing denoising and compression artifacts. Our results demonstrate that patches are not essential to image denoising. Without the additional complexity of patches, our algorithms are surprisingly simple.

Acknowledgments

This work would not have been possible without the support of many people. I would like to thank the following groups and individuals in making this journey possible.

Thanks to Prof. Dr. Matthias Zwicker for accepting me as his first PhD student in the *computer graphics group* (CGG) in Bern, for giving me freedom in my research and tolerating my persistence, for providing reliable advices, technical expertise, support, and trust.

Thanks to all the members of our institute, the *institute of computer science and applied mathematics* (IAM), for providing a cozy environment fit for research.

Thanks to Assistant Dragana Esser, mother of CGG, for bringing warmth into our group, for taking care of all administrative work, for allowing me to focus on research, and for baking all the delicious cakes!

Thanks to all my PhD colleagues. Wan-Yen Lo, for reviewing my paper writing and presentation skills and pushing to aim higher. Fabrice Rousselle, for sharing my interests in rendering and image denoising. Daljit Singh Dhillon, for discussing PDEs and Fourier transforms. Marco Manzi for sharing interests in rendering and demonstrating how effortlessly code can be written. Daniel Donatsch for sharing the office and everyday's happenings and organizing the PhD coffee breaks and pizza events. Peter Bertholet for being a maven in math and provider in brain teasing puzzles.

Thanks to Peppo Brambrilla, for being a reference in system administration.

Thanks to all my masters and bachelor students, for accepting me as a mentor.

Thanks to Prof. Dr. Vladlen Koltun, for allowing me to use the facilities at Stanford University, where this thesis was written.

Thanks to Prof. Dr. Alessandro Foi, co-author of BM3D and many related works, for agreeing to serve as co-referee for this thesis and my defense.

Thanks to Charis Tsevis for allowing me to use his artworks in our papers and this thesis.

Thanks to Patrick Bouchaud and Jarod Tang, for giving me courage to start this endeavour.

Thanks to all the anonymous reviewers of our work.

Thanks to all my friends and relatives, for understanding that research is time consuming.

Last not least, thanks to my parents, for having taught me math from early on, for buying the first home computer, for exposing me to an environment full of music, art, science and culture, for taking me on travels, for supporting me through my studies, for always standing behind me, and for their love.

Contents

Figures

Tables

Algorithms

1

Introduction

Don't let the noise of others' opinions
drown out your own inner voice.

Steve Jobs

What is image denoising? Ideally, this is the question we would like to answer in this thesis. At first, it seems funny to ask this question. After all, this field has existed for half a century and we would expect that such a fundamental question has been answered by now.

Image denoising and digital image processing are indeed classic fields. Ever since images have been digitized and processed, whether in photography, medicine, astronomy, or robotics, image denoising has been the achilles heel of other processing methods. For example in computer vision, the performance of low-level vision and high-level vision tasks depends on the quality of the input image. Seeking the holy grail, the quest for the highest possible image quality has been persued by thousands of researchers in various domains. Consequently, the literature in image denoising is vast and overwhelming. The aspiring researcher has the impression that one has to become an expert or at least have the guidance of an expert before making a difference in this matured field. Today, even experts are pessimistic, asking if the field is dead.

Evidently, progress in image denoising has slowed down. For one thing, the quality of denoising results have come close to theoretical limits, at least for natural images. State-of-the-art image denoisers preserve every perceivable detail while removing most of the noise. Yet, without exception, all of them suffer from artifacts, particularly visible when images depict smooth objects like sky or skin. Figure 1.1 and Figure 1.2 show that state-of-the-art denoising methods introduce artifacts when denoising a simple gradient. Moreover, recent state-of-the-art methods have become complex to implement, making it difficult to further research based on analysis.

With this work, we want to give hope to current and future researchers. Our research shows that surprisingly simple algorithms can achieve and even surpass state-of-the-art results. Most importantly, our methodology has been by and large empiric, and has required little knowledge of the literature. We believe that our results hint towards the existence of simpler theories, eventually answering the stated question.

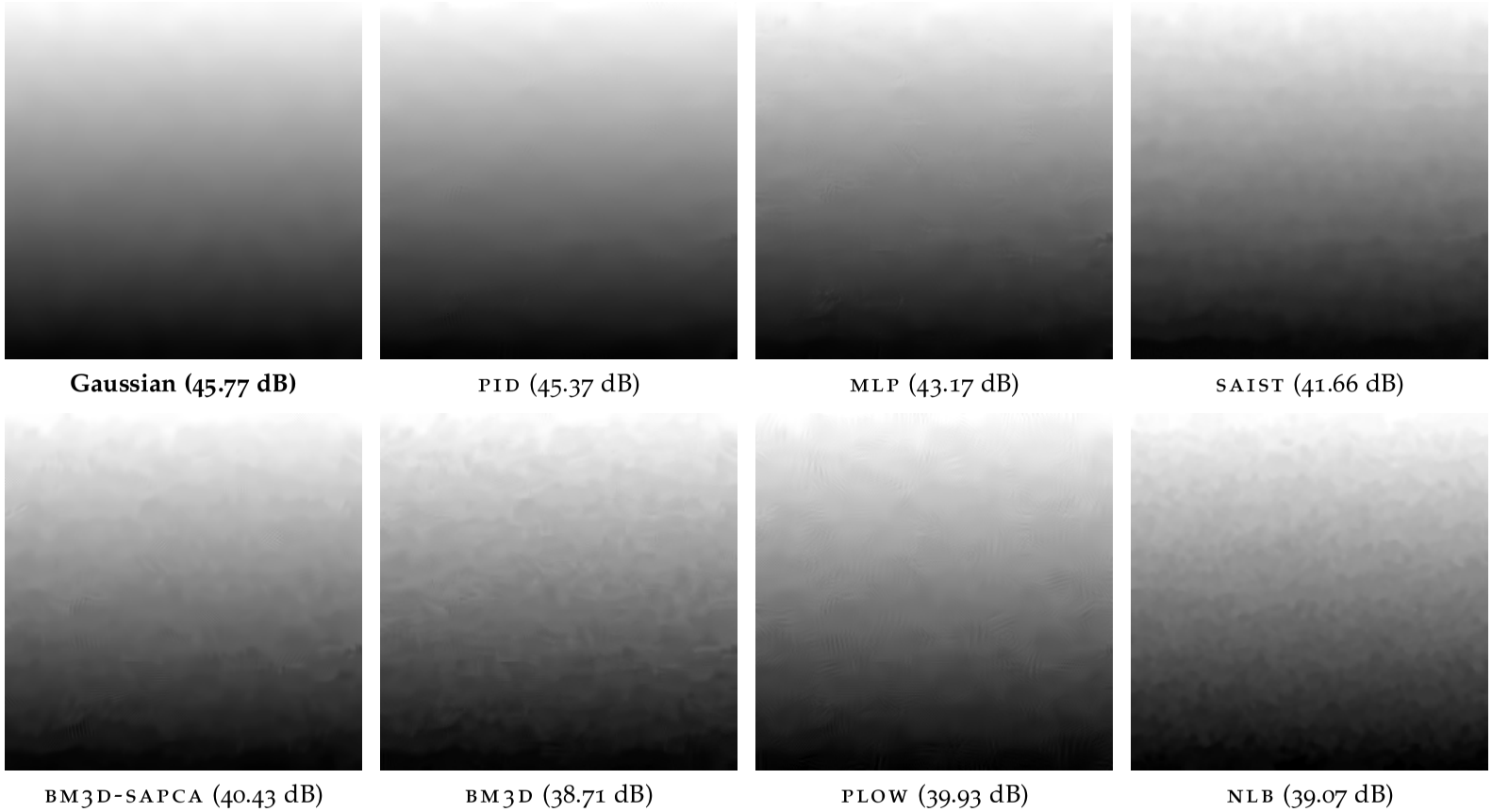


Figure 1.1: *Comparison of artifacts of image denoising methods for noise sigma $\sigma = 25$. Images are ordered by decreasing quality. State-of-the-art methods perform worse than a simple Gaussian for denoising smooth images. One of the methods we propose is* PID*, which produces close to ideal results.*

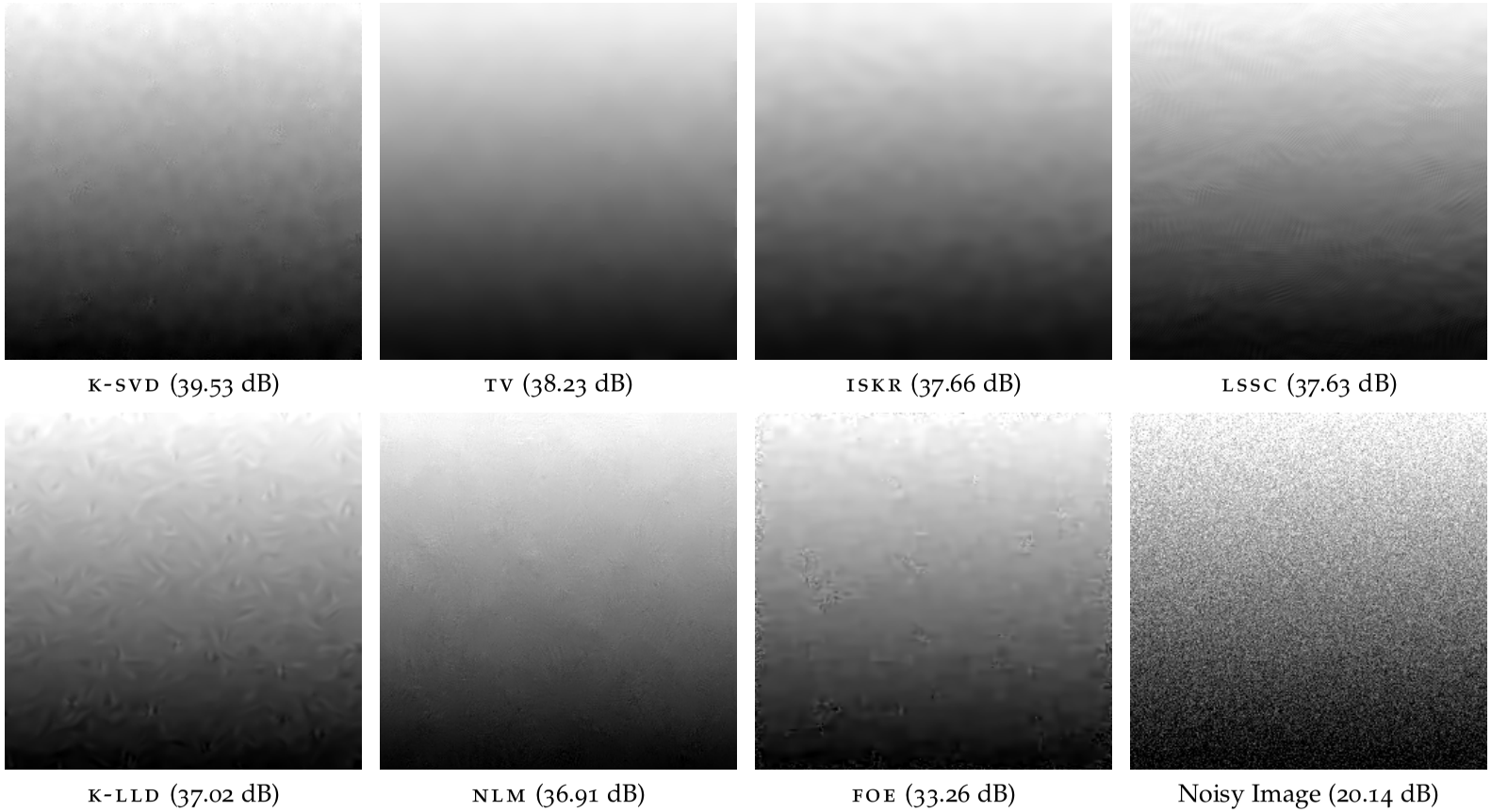


Figure 1.2: *Comparison of artifacts of image denoising methods for noise sigma $\sigma = 25$. The* TV *method looks visually good, but suffers from bias near the boundary.*

y x n

Figure 1.3: *Decomposition of a noisy signal y into its original image x and noise instance n.*

1.1 Image Denoising

While we have no satisfying answer yet, we can at least formulate the classic problem. We consider images as real-value functions defined over a two-dimensional finite discrete domain. The problem of image denoising is to decompose a noise contaminated image y into the sum of its original image x and the noise instance n as

$$y = x + n. \tag{1.1}$$

This equation is also visualized in Figure 1.3. The noise n is assumed to be normally distributed and independent of the original image x. Furthermore, the noise of two different pixels are assumed to be uncorrelated and they have the same known variance σ^2. This type of noise contamination is called *additive white Gaussian noise* (AWGN), inspired by the analogous spectral property of visible white light.

Of course, the AWGN is a simplifying model. In reality, the noise may not be additive, may be correlated with the signal, its distribution may not be Gaussian, and pixel noise may be correlated, and have non-homogeneous variance. Worse, these properties may not even be apriori known.

However, the AWGN serves as a good starting point. In simple cases, the differences in distribution can be ignored as the Gaussian assumption is malleable. Some of the unknown properties can be found in a pre-processing step. A basic assumption is that the variance is known, which is not true for any real image given in isolation. In response, many techniques for variance estimation have been developed. In other cases, the noisy image y can be transformed to meet the assumptions using *variance stabilizing transformations*

y $\tilde{x}$ $\tilde{n}$

Figure 1.4: *An example of denoising a noisy image y, resulting in a denoised image $\tilde{x}$ and method noise $\tilde{n}$.*

(VST). An example is to transform multiplicative noise into additive noise by taking the logarithm. Another example is the application of the Anscombe transform to turn Poisson and Rayleigh distributions into a normal distribution [1]. For more specific cases, a generic AWGN based algorithm should be extended to handle the specific noise model.

Given that the AWGN assumptions are valid, the problem is still considered "ill-posed". In general, information is lost, making it impossible to reconstruct the original image x. In the simplest case, the original image is a constant function, allowing near perfect reconstruction by mere averaging neighbouring pixels. In the hardest case, the original image itself is pure white Gaussian noise. Since the original pixels of the image x and therefore the noisy image y are uncorrelated, neighboring pixels do not contain any additional information and the best estimation of x is y. Therefore, interesting images which contain some randomness are subject to loss of information.

In practice, the ideal decomposition can only be approximated, and any denoising method decomposes the noisy image y into the denoised image $\tilde{x}$ and the method noise $\tilde{n}$ as

$$y = \tilde{x} + \tilde{n}. \tag{1.2}$$

Visually, it will look as shown in Figure 1.4. Instantly, the question for the best possible image arises. Assuming the image x^* would be statistically the best possible approximation of a given x and noise instances n with variance σ^2, how could we define it?

It is sobering that no such definition has been found in the literature. If such a definition existed, image denoising could be considered "solved", in the

sense that it remains an engineering exercise to seek efficient or elegant ways to calculate the best image x^*. Researchers pursuing the approach of total variation (TV) have the ambition to find such a definition, as they define ad hoc objective functions to optimize. The results have been so far underwhelming, letting us conclude that the correct objective function has not been found yet either. Another popular approach is based on patches where, given a noisy patch, the most likely patch is seeked. Approaches based on neural networks work in principle, but require expensive off-line learning and do not give us insights about image denoising. Others define objective functions over the patches, but even those approaches do not produce entirely satisfying results. Today, researchers and their methods are still competing to find closer approximations to the best image x^*. We join the competition and take the challenge in the quest of the holy grail.

1.2 Contributions

The contents of this thesis stems from three publications in the area of image denoising. The first publication is a conference paper that carries the same title as our thesis, *Dual-Domain Image Denoising* (DDID), which was presented at the IEEE *International Conference on Image Processing* (ICIP) 2013 and won the student award [2]. This work introduces the main idea of our thesis, to perform image denoising in both spatial and frequency domains. It distinguishes itself from other hybrid approaches in that both spatial and frequency domain operations are the same. This work stands out as it demonstrates that state-of-the-art denoising can be performed by a very simple algorithm.

The second publication is a journal article called *Progressive Image Denoising* (PID) submitted to IEEE *Transactions on Image Processing* (TIP) [3]. At the time of writing, it was under review. PID improves over DDID by producing denoised images with nearly no artifacts. This is convincingly demonstrated by denoising synthetic images, where edges, gradients, and homogeneous areas are abound.

The third and last publication is a conference paper called *Dual-Domain Filtering* (DDF) and was just submitted to IEEE *Computer Vision and Pattern Recognition* (CVPR) 2014 [4]. DDF generalizes the filtering operation at the core of DDID and PID and introduces it as a stand-alone tool. We show that DDF is very potent in removing residual noise from other image denoising methods, as well as removing compression artifacts. Furthermore, using the DDF, we present an improved image denoising method, which combines the strengths of DDID and PID, simplicity and quality, into a coherent method named DDID2.

The denoising quality of DDID2 is numerically and visually on par with the best methods for denoising grayscale images, and, in average, beats all known methods for color images.

We summarize our practical contributions:

- We propose DDF, a new image filter that recasts the bilateral filter as a robust noise estimator and generalizes to include the frequency domain. Thanks to this generalization, our filter not only preserves edges, but also details. Our filter is powerful in removing artifacts like residual noise and compression artifacts.
- Based on DDF, we propose three denoising methods, DDID, PID, and DDID2. All of them achieve state-of-the-art results, while being exceptionally simple to to implement. PID and DDID2 produce nearly artifact-free results, making them suited for denoising synthetic images, and they also surpass the state-of-the-art in denoising color images.

Our results lead to new insights in image denoising:

- Our methods show that state-of-the-art image denoising can be achieved without using patches.
- Our methods also show that learning statistical models of the signal is unnecessary. We know the statistics of the noise, i.e., the variance, which is sufficient to denoise AWGN contaminated images by robust noise estimation.
- The implementation of PID is based on deterministic annealing, which hints at image denoising being a physical process.

Note that in the literature, *noise estimation* usually refers to the estimation of the statistics of the noise, such as the first and second moments of the noise. In this thesis, we will use the term *noise estimation* to mean the estimation of the noise realization, or noise instance. Because of its frequent use, we will henceforth use the term noise estimation.

1.3 Overview

The chapters of this thesis are grouped into three parts. The opening part consists of this introduction, our methodology in Chapter 2, and related works in Chapter 3.

The main part starts by introducing the foundation of this thesis, the *dual-domain filter* (DDF) in Chapter 4. We elaborate three generations of image

denoisers based on this filter. First, *dual-domain image denoising* (DDID) in Chapter 5, then *progressive image denoising* (PID) in Chapter 6, and a combined method (DDID2) in Chapter 7. We also demonstrate additional applications of DDF for artifact removal in Chapter 8.

The closing part first discusses the implementations of DDF, DDID, PID, DDID2, and artifact removal in Chapter 9. Then, the results are shown using images, plots, and tables in Chapter 10. Finally, we conclude our work in Chapter 11.

2
Methodology

Philosophy of science is about as useful to scientists as ornithology is to birds.

Richard Feynman

How to attack a classic problem like image denoising? Common sense tells us to first study the literature, then come up with an alteration or combination of exisiting methods. In the case of image denoising, the sheer size of the literature is daunting. Many methods exist, seemingly pointing into different directions.

Even if the student somehow attains expert level knowledge, how to differentiate from the existing elite? Given the same knowledge on the subject, what is the secret in coming up with something new, somebody else has not found already?

2.1 Scientific Method

The answer lies not in the theory, but in the practice. Everybody in academia has access to the same knowledge, but personal experience is unique. The best way for a researcher to differentiate from the crowd is to perform experiments. Based on experiments, the attentive researcher makes observations, which eventually lead to forming hypothesis, ideas, or even theories. The new hypothesis is subject to another round of experimental validation. This cycle is known as the scientific method.

Unfortunately, experience tells us that most of our ideas and hypothesis are wrong. It is therefore important that our ideas can be formulated in form of questions which can be answered by an experiment, and in most cases to be rejected.

Today, we are in the best position to perform empiric research, especially for sciences that are home in the digital domain. Hardware in form of multi-

core CPUs, GPUs, and clusters of them offer access to exponentially growing computational power. Combined with high-level software tools like MATLAB allow for rapid prototyping and thus accelerated research.

2.2 Genetic Algorithm

In practice, for this research, we took the approach of the evolution inspired genetic algorithm. We started with a population of denoising algorithms and used a metric with a set of images to evaluate the fitness of each algorithm. For every method, we first fitted parameters until every method produced the best possible results. When the population grew, we performed crossovers, by combining complementary methods to create hybrids which would perform better. Alternatively, when the population shrunk, we performed mutation by changing an algorithm in an arbitrary way. Iterating this procedure eventually lead to the algorithms presented in this thesis.

An important deviation from the standard genetic algorithm is that every step was manually taken, even parameter optimization. This was critical as collecting more observations increases the chance of getting new ideas. We also used refactoring to find simpler algorithms without altering the population. The presented algorithms are deemed by the author as simple as possible.

2.3 Metrics

We use the most common ways to gauge the quality of denoised images, the *peak signal-to-noise ratio* (PSNR) and the judgment of our own eyes. In this section, we first describe the most basic metric, the *mean squared error* (MSE), and then its derived and more popular metric, the PSNR. We also discuss the *structural similarity index metric* (SSIM) and the reasons why we chose against it.

2.3.1 Mean Squared Error

The *mean squared error* (MSE) is the most basic error metric for images. As the name suggests, the MSE calculates the squared difference between a test image $\tilde{x}$ and a reference image x. However, the term *mean* can be understood in two ways. The MSE is a statistical measure, which averages either over time or over space. When we consider a particular pixel, the MSE is the expected error of the pixel. This value cannot be calculated from a single image, but only by averaging for the particular pixel the squared error over a collection

of images $\tilde{x}_i$ with the same statistics. What is more common and useful for image processing rather than averaging over time is to average over space. The squared error is averaged over all pixels p of an image, which is how usually the MSE is calculated for images, i.e.,

$$MSE = \frac{1}{wh} \sum_p (\tilde{x}_p - x_p)^2, \tag{2.1}$$

where w and h are the width and height of the image in pixels.

2.3.2 Peak Signal-to-Noise Ratio

In the image denoising literature, the more convenient metric *peak signal-to-noise ratio* (PSNR) dominates. It can be calculated from the MSE as

$$PSNR = -10 \log_{10} \frac{MSE}{MAX^2} \tag{2.2}$$

While the PSNR does not contain more information than the MSE, it is more convenient in many ways. First, it normalizes the error by the squared maximum possible value MAX^2. This makes the error scale independent. Most images are stored using 8-bits per channel resolution, allowing the maximum value $MAX = 255$. Other images may use less or more bits of range resolution, or even store floating point values as used by HDR images. For practical purposes, an image normalized between 0 and 1 can be more intuitive. A second benefit of the PSNR is its transformation into log scale. Weber's law states that human vision perceives exponential light steps as linear. The log transformation therefore alignes the metric closer with human perception. A third benefit is the use of the negative log 10 base, which scales the PSNR such that better images have larger values. Last not least, PSNR numbers are in a range which are easy to memorize and relate to. Typical PSNR values are below 50 dB and for all purposes, two digits after the period are sufficient to distinguish the quality of denoised images.

2.3.3 Structural Similarity Index

Recently, another metric has gained popularity among some researchers, the *structural similarity index metric* (SSIM) [5]. The SSIM of a pixel is defined over its neighborhood of pixels as

$$SSIM(x, y) = \frac{2\,\mu_x\,\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \frac{2\sigma_{xy} + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}, \tag{2.3}$$

where x and y are the two images being compared. The sample means $\mu_{x,y}$, the sample variances $\sigma^2_{x,y}$, and the sample covariance σ_{xy} are statistics calculated over the neighborhood of the considered pixel. The constants c_1 and c_2 avoid singularities in dark and flat regions. To compute the metric for an entire image, the SSIM value is averaged. Its definition is

$$MSSIM(x,y) = \frac{1}{wh}\sum_p SSIM(x_p, y_p). \tag{2.4}$$

This metric is called MSSIM, but often the "M" is dropped.

Horé and Ziou have shown that for most degradation types, a simple mathematical relationship holds beween the MSSIM and PSNR [6]. We think therefore that PSNR as a single objective metric is sufficient for objectively quantifying image denoising results.

2.3.4 Visual Assessment

Unfortunately, objective metrics are insufficient to describe the human perception of a noise free image. Perceptual metrics like the state-of-the-art HDR-VDP-2 [7], are not well suited for assessing image denoising results. For example, when the original image has already noise or consists of stochastic patterns, a method which produces residual noise may get a good score because both images are noisy, even though the two noise patterns are uncorrelated. Without a reliable objective perceptual metric, we resort to trusting the author's *human visual system* (HVS).

3
Related Works

We don't see things as they are.
We see them as we are.

Anais Nin

In this chapter, we review previous works related to our contributions. We organize this chapter by the most important paradigms in image denoising. In Section 3.1, we first look at image denoising as a solution to an energy minimization problem. After that, we describe the classic methods that denoise either in the spatial domain (Section 3.2) or in the wavelet domain (Section 3.3). In Section 3.4, we move on to the currently dominating paradigm, the denoising of image patches. We discuss the various methods that describe patches using dictionaries and how these dictionaries are obtained. Patches are also attractive as they allow for theoretical analysis. Finally, in Section 3.5, we discuss the connection of the literature to our work.

3.1 Energy Minimization

The wishful thinking for image denoising is that we could formulate a smooth and convex energy function over the entire image or over a limited neighborhood for every pixel. If that was possible, we could straightforward minimize this energy with a gradient descent. So far, no such energy formulation has been found that denoises images well. We describe the past denoising attempts which admit an energy based formulation.

3.1.1 Anisotropic Diffusion

A classic energy formulation is

$$E = \int_{\mathbb{R}^2} k\left(|\nabla y(p)|^2\right)\, dp. \tag{3.1}$$

The energy E is simply the energy of the gradient over the whole image. If the function k is the scale operator, the gradient descent formulation of this energy minimization is the well known heat equation, also known as diffusion equation. Every time step corresponds to a convolution of the noisy image with an enlarging Gaussian kernel. The Gaussian kernel is the most basic image denoiser and works only well for nearly homogeneous images. This is consistent with the fact that the (steady state) solution to this equation is a constant.

If the gradient of the function k is a robust kernel, e.g., a Gaussian or a Lorentzian, we get instead the *anisotropic diffusion* (AD) as introduced by Perona and Malik [8]. The robust estimator ∇k becomes smaller for large gradients end prevents diffusion across edges, thus preserving such features. Black et al. have studied anisotropic diffusion using alternative robust kernels [9].

The AD formulation is attractive: while the energy is globally formulated, the update operations only reguire the local gradient. Unfortunately, using the proposed diffusion coefficients, details are washed out and the image converges to a staircase signal.

3.1.2 Bilateral Filtering

In its original form, anisotropic diffusion was considered as a strictly local method, but energy functionals over a neighborhood of pixels have been considered, which can be used to define the *bilateral filter* (BF) introduced by Tomasi and Manduchi [10]. The bilateral filter calculates the bilaterally filtered value $\bar{x}_p$ for pixel p using pixels q in the neighborhood $\mathcal{N}_p$ as

$$\bar{x}_p = \sum_{q \in \mathcal{N}_p} y_q\, k_q \Big/ \sum_q k_q, \tag{3.2}$$

where the bilateral kernel k_q is defined as a product of two Gaussians as

$$k_q = e^{-\frac{(q-p)^2}{2\sigma_s^2}}\, e^{-\frac{(y_q - q_p)^2}{\gamma_r\, \sigma^2}}. \tag{3.3}$$

The left term is the spatial Gaussian with σ_s being the standard deviation. The right term is the range Gaussian with γ_r being a scale factor for the noise variance σ^2.

Although initially presented as an ad hoc method, the bilateral filter can be found as the first Jacobi step in minimizing an energy defined by gradients over a neighborhood window [11]. Its connection to anisotropic diffusion has also been mentioned in [12, 13].

Bilateral filtering works best for low noise situations. More noise can be removed by iterating the filter, but this leads to staircasing, just like anisotropic diffusion [14]. Nevertheless, bilateral filtering has proven itself as an all-round edge-preserving editing tool. Adaptive methods have been implemented for HDR compression, detail enhancement, and JPEG deblocking [15, 16, 17].

3.1.3 Total Variation

The best known attempt to formulate image denoising as a variational problem was formulated by Rudin et al. [18]. The energy minimization problem they propose is

$$E = \int_{\mathbb{R}^2} |\nabla x(p)|\, dp + \lambda \int_{\mathbb{R}^2} (y(p) - x(p))^2\, dp, \tag{3.4}$$

also called *Rudin-Osher-Fatemi* (ROF) problem. The energy consists of a regularization term and a data term. The regularization term is the total variation norm, which is an L_1 norm over the gradient, rather than a squared L_2 norm as in Equation 3.1. The data term prevents the solution to become flat as in the case of the heat equation. The advantage of using a L_1 norm is that it leads to sparse signals, that is, the solution will be mostly flat and contain some steps. However, minimization of L_1 is not as straightforward as the L_2 norm. Besides the difficulty in finding the solution, depending on the parameter λ, the resulting denoised images make a trade-off between residual noise and loss of detail.

3.2 Kernel-based Methods

Classic image denoisers are formulated as filters using a convolution with an adaptive kernel. The most popular filter is the previously mentioned bilateral filter. Another adaptive method is *steering kernel regression* (SKR) [19], where the bilateral kernel is used to fit an anisotropic Gaussian.

Both methods work better for low-noise situations, as for high-noise situations details are washed out. A method which preserves more detail is *non-local means* (NLM) by Buades et al. [20]. NLM is an extension to the bilateral filter, where the distances between two pixels are replaced by distances betweeen patches. Details are better preserved by detecting self-similarity between patches.

The popularity of NLM is due to the combination of its simplicity, the existence of efficient implementations, and the potential to produce visually pleasant results. The NLM method discussed is the pixel-based NLM. As we will discuss later, NLM can also be implemented as a patch-based variant, which may be a reason for the boom of patch-based methods.

3.3 Wavelet-based Methods

An entire class of denoising methods are based on wavelet transforms [21, 22, 23]. They transform the noisy signal into the wavelet domain, and operate on the obtained wavelet coefficients based on the noise statistics. The three common types of operations are hard thresholding, soft thresholding, and Wiener filtering. Thresholding implies a hard classification of each coefficient into signal and noise, based on the amplitude of the coefficient. Hard thresholding reduces coefficients with amplitudes smaller than the threshold to 0, while keeping the signals with amplitudes larger than the threshold. Soft thresholding subtracts a threshold value from all amplitudes and resulting negative coefficients are clamped to 0. Extensive research has been done for finding good threshold values [21, 22, 24]. Wiener filtering makes no hard classification, its factors vary smoothly from 0 to 1. The Wiener filter is popular due to its MSE minimizing property. However, this is a theoretical result, since it requires an oracle that has to be approximated.

Sometimes, the term *shrinkage* and *soft thresholding* are used interchangeably in the literature. However, shrinkage is the general term for any kind of attenuation of spectrum coefficients, including hard and soft thresholding and Wiener filtering.

3.4 Patch-based Methods

Recently, the focus has shifted from denoising entire images and single pixels to image patches. The general framework of patch-based methods is as follows. For every pixel, the enclosing patch is denoised, using other patches in its

neighborhood. The final step is called aggregation, where the denoised patches are accumulated into the image buffer, which is then normalized to handle overlapping patches.

The key idea of using patches as a denoising primitive is that statistics over patches can be collected, allowing to estimate the patch with the maximum likelihood. A direct approach has been taken by Buades et al. in *Non-Local Bayes* (NLB) and by Chatterjee and Milanfar in *patch-based locally optimal Wiener* (PLOW).

The basic assumption underlying patch-based methods is that patches are normally distributed and the distribution is centered around the noise free patch. The Achilles heel of patch based methods is to find enough patches that belong to the same distribution, i.e., patches which correlate with the patch to be denoised. Intuitively, patches that are further away are less correlated than closer patches. Levin et al. have analyzed that pixels across edges are almost uncorrelated [25]. The common way to distinguish correlating patches from non-correlating ones is to use patch distances based on the L_2 norm, and to perform a threshold either on the distance or on the number of patches [26, 27]. Other methods separate patches by clustering [28, 29].

NLM is possibly the most basic patch-based method. Besides the pixel-based variant, NLM can be also implemented patch-wise. A subset of the image pixels are chosen as centers of overlapping patches, such that the entire image is covered by them. For every patch location, the patches are bilaterally filtered by using the squared norm of the vectorized patches. Finally, every pixel is calculated by averaging the denoised patches containing the pixel.

If the region surrounding the patch to be denoised is dominated by strong features like edges and corners, it may be difficult to find enough correlating patch samples. For example, when the patch to be denoised covers a strong feature like an edge, the number of correlating patches are limited to the pixel positions in the direction of the edge. The simplest way out is to avoid the edge by decreasing the patch size at the cost of losing details or introducing noise. A more sophisticated way to increase the number of patches without reducing the patch size is to use data-adaptive shapes. Shape-adaptive methods like SA-BM3D and BM3D-SAPCA [30, 31] mask out the patch regions with strong features. The shape is given by a polygon mask, centered around the center pixel. A similar path is taken by K-LLD, where patches are grouped using SKR.

3.4.1 Sparsity

Instead of rigorously finding the most likely patch, most methods are more heuristic and are based on sparsity. The idea here is that the noise-free patch can locally be represented as a linear combination of a few functions, called atoms. The set of all atoms is called dictionary, which can be complete or overcomplete. Dictionaries are overcomplete if the number of atoms exceeds the dimensionality of the signal and, therefore, the atoms linearly depend on each other. A property of overcomplete dictionaries is that they allow for sparser representations.

The noisy patch is denoised by *sparse coding*, the process of finding a sparse representation using the given dictionary. For complete dictionaries, usually based on wavelets, thresholding, or Wiener filtering is used. In the latter case, sparsity is not strictly enforced in an L_0 sense, as the wavelet coefficients are multiplied by non-zero factors. For overcomplete dictionaries, greedy heuristics called pursuit methods find near-optimal sparse representations. While producing numerically good results, wavelet methods are prone to disturbing ringing artifacts, which is a reason why overcomplete dictionaries are preferred.

There are many strategies to obtain a good dictionary. The simplest way is to use a fixed dictionary, like complete or overcomplete wavelets. A well-known method using a fixed dictionary is BM3D. However, dictionaries can also be learned from data. Some methods process image databases off-line, for example [32]. A method that learns on-line over the whole image is K-SVD by Aharon et al. [33]. K-SVD is a generalization of the *k*-means algorithm, which iteratively performs sparse coding and dictionary update. The difference to *k*-means algorithm is that the sparse encoding is not limited to a single atom and can have weights other than one. More recently, dictionaries are learned over clusters (K-LLD, BM3D-SAPCA), to make use of *simultaneous sparse coding* (SSC).

The idea of SSC comes from the observation that patches which are similar have stronger correlation than patches which are different. If two patches are strongly correlated, then their sparse codings should also be correlated. For complete dictionaries, the most known method making use of SSC is BM3D by Dabov et al. [26], which stacks and transforms patches in three dimensions, the two local dimensions and a third non-local dimension. Similarly, for overcomplete dictionaries, the same structural sparsity can be enforced over all patches. This idea was taken by LSCC, proposed by Mairal et al. [34]. BM3D-SAPCA by Dabov et al. [31] performs a *principal component analysis* (PCA)

over the set of patches, thus making use of non-local similarity between patches. A related approach was taken by Chatterjee and Milanfar who proposed K-LLD, using locally learned dictionaries for each cluster of patches.

A new path was taken by Dong et al. who recently proposed *spatially adaptive iterative singular-value thresholding* (SAIST). They replace sparse linear combination using a single dictionary by a *singular-value decomposition* (SVD) with two dictionaries. Their formulation makes the effect of sparsity symmetric in row and column space, with the orthogonal matrices of the decomposition corresponding to local and non-local dictionaries.

3.4.2 Alternative Patch-based Methods

A deviation from the common patch-based methods is a method based on multi-layer perceptrons (MLP) by Burger et a. [35]. Instead of learning dictionaries, MLP learns the direct mapping from noisy image to the denoised signal itself. An advantage of this method is that it can learn any degradation type. The downside of this off-line learning method is that it takes many GPU-days to learn for every degradation type, including a simple change in the noise variance.

Another example of a off-line learning method is *Field of Experts* (FOE). It also learns a dictionary off-line, but uses probabilistic inference to find the most likely image using a prior on the entire image. Like the MLP, its framework is also flexible enough to generalize to other degradation types than denoising.

3.4.3 Theoretical Limits

With the rise of patch-based methods, denoising quality has improved so much that theoretical limits of denoising have become a topic. Levin and Nadler [36] and Chatterjee and Milanfar [37] analyze these limits based on patches and come to the conclusion that for natural images, methods like BM3D are close to what is theoretically possible. They also conclude that denoising smooth images still has potential for improvement.

3.5 Discussion

We discuss how our work contrasts from recent development of the field. We note that all state-of-the-art methods are based on patches. Our methods do not follow the trend and are based on the classic works of bilateral filtering and short-time Fourier transform (STFT).

Most patch-based methods first search for correlating patches. In our approach, we keep the entire search window, so we do not need to locate correlating patches. We also do not have to use shape-adaptive methods to increase the number of correlating patches. We simply use the bilateral filter to discard uncorrelated pixels. Methods like BM3D and SAIST still treat self-similarity of patches, i.e., local and non-local correlation, distinctly. In our world, both local and non-local self-similarity translate simply to auto-correlation of the search window, i.e., waves.

It seems plausible that if we have explicit statistical knowledge about the signal, we can improve denoising. However, this has not be proven yet. Patch-based methods [38, 26, 31, 27, 34, 29] collect statistics over patches and use the sparsity argument to denoise the patches. Our methods do not require learning of dictionaries, as we use simple Gabor wavelets. Learned dictionaries are statistical models of the signal. In our works, we make no statistical assumptions about the signal. Instead of trying to collect statistics about the signal, we focus on the existing statistical knowledge about the zero-mean white Gaussian noise, which is given by its variance. Using the variance, we perform robust noise estimation by discarding large signals as outliers and estimate the noise. In other words, instead of estimating an unknown model of the signal, we use the known noise model and discard signals which do not fit the model.

Given that our patch-less denoising methods directly compete with the state-of-the-art, we make the case that patches and the tools developed for them are unnecessary for image denoising.

We note that for a long time, the image denoising community has been focused on denoising natural images. Given the traditional application domains of denoising, this may not be surprising. Consequently, a lot of effort is put into collecting statistics about natural images and how to exploit them for image denoising [36]. Synthetic images have not gotten the same attention as natural images. Current image denoising methods produce unacceptable artifacts for synthetic images. In our work, we show that no distinction has to be made between denoising natural and synthetic images. Since we do not use any statistics of the signal, our methods are signal agnostic.

Finally, we also note that there is a lack of contribution for denoising color images. Only BM3D and NLB count as state-of-the-art methods for denoising color images. Other methods have only shown their performance for grayscale images. It is not entirely clear to the author if lack of interest or technical difficulties are the reason for not extending current methods to color images and more generally to hyper-spectral images. However, our algorithms presented in

this work produce visually and numerically the best color images.

4

Dual-Domain Filter

What is a scientist after all?
It is a curious man looking through a keyhole,
the keyhole of nature, trying to know what's going on.

Jacques-Yves Cousteau

Image enhancement and reconstruction are important tasks in image processing. Images may be degraded by AWGN, by arbitrary method noise or compression artifacts. To improve such images, specialized tools are often developed for each type of degradation.

Some image processing tools are generally potent to attack such problems. The bilateral filter (BF) [10] and its variant, the joint-bilateral filter [39], have become popular tools due to their simplicity and effectiveness in removing named artifacts. For example, bilateral filtering can be used for denoising images contaminated with weak noise or for removing unwanted details. Also, adaptive bilateral filtering has been proven effective for JPEG deblocking by Zhang and Gunturk [17] and Nath et al. [40].

However, the bilateral filter, due to its spatial definition, makes a trade-off between removal of noise and loss of contrast and detail. Typically, details are better preserved using transform domain methods, which is why some JPEG deblocking methods inspect the DCT coefficents of the blocks. Sophisticated image denoising methods operate in both spatial and frequency domains. Moreover, it is common to cast artifact removal as a denoising problem, by simply using existing denoising methods [38, 19, 20, 41, 26, 28]. However, the best denoising methods are complex to implement and are not part of every image processing engineer's toolbox like the common bilateral filter.

In this chapter, we introduce a simple but powerful image processing filter that we call *Dual-Domain Filter* (DDF). With DDF, we flip the semantics of bilateral filtering and wavelet shrinkage and combine them as robust noise

estimation in two domains. Moreover, we offer an extension to allow guided filtering using a second image. Recall that, in this thesis, we use the term *noise estimation* to mean the estimation of the actual noise instance, rather than the statistics of the noise which are known.

We formulate the DDF as a robust noise estimator in two domains. Typical image denoising filters estimate a signal x directly from a noisy input y, attempting a decomposition $y = x + n$, where n is the noise. In contrast, our filter first estimates the noise n which is then subtracted from the noisy signal y to obtain x. This seemingly subtle difference matters when estimating the noise in multiple domains. While we make no assumptions about the signal, we assume to know the noise statistics. The noise statistics are used to robustly estimate the noise first in the spatial domain, then in the frequency domain.

The DDF first uses the bilateral filter to estimate the noise $\bar{n}_p$ in the pixel value y_p using neighborhood pixels $q \in \mathcal{N}_p$, limited by kernel radius r. It then reestimates the noise $\hat{n}_p$ in the frequency domain using the frequencies $f \in \mathcal{F}_p$. The noise estimations in the two domains are described in Section 4.1 and Section 4.2. In Section 4.3, we show a way to let DDF be guided by a second image. We discuss our dual-domain robust noise estimation paradigm in Section 4.4 and give a short outlook on the following chapters in Section 4.5.

4.1 Noise Estimation in Spatial Domain

The spatial domain filter is a reformulation of the bilateral filter, now designed to estimate the noise $\bar{n}_p$. We first subtract the pixel y_p from the neighbor pixels y_q, forming the "gradient" d_q as

$$d_q = y_q - y_p. \tag{4.1}$$

Then, we use the squared norms of the gradient d_q and the distance $q - p$ to define the bilateral kernel k_q as

$$k_q = k\left(|d_q|^2, |q-p|^2\right), \tag{4.2}$$

where the bilateral kernel function $k(\cdot,\cdot)$ is assumed to be a robust kernel that normalizes the squared norms according to the known noise statistics. In particular, the kernel function is responsible for spatial and range scale parameters, for example as $k(|d_q|^2, |q-p|^2) = e^{-\frac{|d_q|^2}{\gamma_r \sigma^2}} e^{-\frac{|q-p|^2}{2\sigma_s^2}}$. This control over the spatial and range scale parameters is critical for iterating DDF as it will be shown in in the following chapters.

Finally, we estimate the noise $\bar{n}_p$ by locally convolving the gradient d_q with the normalized bilateral kernel $k_q / \sum_{q \in \mathcal{N}_p} k_q$:

$$\bar{n}_p = a \sum_{q \in \mathcal{N}_p} d_q \, k_q \Big/ \sum_{q \in \mathcal{N}_p} k_q. \tag{4.3}$$

We additionally introduce a confidence factor a ranging from 0 to 1. If we have no confidence in the noise estimate, the confidence value is $a = 0$. If we have full confidence, the confidence value is $a = 1$. This factor is useful for implementing PID and DDID2, which iterate this filter and the confidence value grows over time. The used confidence values are described in the respective chapters. If not otherwise stated, we assume $a = 1$.

The estimated noise $\bar{n}_p$ by Equation 4.3 is equivalent to the bilaterally filtered value $\bar{x}_p$ given by Equation 3.2 minus the noisy value y_p.

4.2 Noise Estimation in Frequency Domain

Similar to the spatial domain filter, the frequency domain filter is a reformulation of wavelet shrinkage, redesigned to estimate the noise $\hat{n}_p$.

We use the previous results to remove bias from the frequency domain noise estimation in two ways. We first remove the spatially estimated noise $\bar{n}_p$ from the gradient d_q. This corresponds to the removal of the DC component under the bilateral kernel, asserting that noise has zero mean. Then, we mask the resulting signal using the bilateral kernel k_q, removing large gradients which otherwise would bias the spectrum. Now we are ready to multiply with the Fourier kernel to take the *discrete Fourier transform* (DFT), yielding the Fourier coefficients D_f as

$$D_f = \sum_{q \in \mathcal{N}_p} (d_q - \bar{n}_p) \, k_q \, e^{-i \frac{2\pi}{2r+1} f \cdot (q-p)}, \tag{4.4}$$

with frequencies $f \in \mathcal{F}_p$. Next, we define the frequency domain kernel K_f to reject correlating signals as outliers. Since the variance of a scaled signal is proportional to the squared factors, we normalize the energy of the Fourier coefficients by the energy of the bilateral kernel k_q. Our definition for the frequency domain kernel becomes

$$K_f = K \left(|D_f|^2 \Big/ \sum_{q \in \mathcal{N}_p} k_q^2 \right). \tag{4.5}$$

Similar to the bilateral kernel function, the frequency domain range kernel function $K(\cdot)$ is assumed to be robust and to normalize the energy according to the known noise statistics.

Finally, we reconstruct the center pixel noise $\hat{n}_p$. We perform the inverse Fourier transform of the scaled Fourier coefficients $D_f K_f$, followed by the evaluation at pixel p by convolving with the Dirac delta function, yielding

$$\hat{n}_p = A \sum_{q \in \mathcal{N}_p} \delta(q-p) \frac{1}{(2r+1)^2} \sum_{f \in \mathcal{F}_p} D_f K_f e^{i \frac{2\pi}{2r+1} f \cdot (q-p)} \tag{4.6}$$

$$= \frac{A}{(2r+1)^2} \sum_{f \in \mathcal{F}_p} D_f K_f \sum_{q \in \mathcal{N}_p} \delta(q-p) e^{i \frac{2\pi}{2r+1} f \cdot (q-p)} \tag{4.7}$$

$$= \frac{A}{(2r+1)^2} \sum_{f \in \mathcal{F}_p} D_f K_f. \tag{4.8}$$

We observe that we can omit the inverse Fourier transform and the noise estimate reduces to a dot product in the frequency domain between the Fourier coefficients D_f and the frequency kernel K_f. The normalization factor $1/(2r+1)^2$ corrects for the fact that the DFT is non-unitary. The parameter A is another confidence factor between 0 and 1. Now we have the final noise estimate $\hat{n}_p$ and we can subtract it from the noisy pixel y_p to get the estimate

$$\hat{x}_p = y_p - \hat{n}_p. \tag{4.9}$$

4.3 Guided Dual-Domain Filter

We can also formulate DDF as a "guided filter", similar to the joint-bilateral filter [39] and the guided image filter [42]. Instead of having a single input image, we have an additional guide image g that defines the filter, which is then applied to the noisy input image y. Since the same computations are performed on both images, we can use a trick by using the complex substitution

$$y \to g + i\, y. \tag{4.10}$$

We only have to make two adaptations. First, we extract the real part as the guide to define the bilateral kernel and Equation 3.3 becomes

$$k_q = k\left(|\operatorname{Re} d_q|^2, |q-p|^2\right). \tag{4.11}$$

Second, the Fourier transform now computes two real Fourier transforms simultaneously, one for the guide image g and another for the noisy image

y. This is a common trick [43] to reduce the complexity of two real Fourier Transforms. We can derive the rule to extract the Fourier transform of the real part from the properties of a real Fourier transform [44]. The real part of the Fourier coefficients of a real Fourier transform is even and the imaginary part is odd. We can therefore extract the Fourier coefficients of the real part as the guide with $\frac{D_f + D^*_{-f}}{2}$ and Equation 4.5 becomes

$$K_f = K\left(\left|\frac{D_f + D^*_{-f}}{2}\right|^2 \Big/ \sum_{q \in \mathcal{N}_p} k_q^2\right). \tag{4.12}$$

Finally, the estimated noise is substituted as $\hat{n}_p \to \operatorname{Im} \hat{n}_p$ and $\hat{x}_p = \operatorname{Im}(y_p - \hat{n}_p)$.

4.4 Robust Noise Estimators

Our robust noise estimation paradigm unifies spatial and frequency domain denoising. Durand and Dorsey have already made the connection between bilateral filter and robust estimation [13]. The bilateral filter is robust in three dimensions, two spatial and one range dimension for the spatial domain. Our estimator adds another range dimension for the frequency domain. Since both domains induce bias, our robust noise estimator protects from bias in both domains.

The original bilateral filter is defined using the Welsch estimator for both spatial and range dimensions. Durand and Dorsey considered alternative robust estimators like the Lorentz and Tukey estimator. To add even more flexibility, it is possible to pick different robust estimators for every dimension individually.

For example, we can consider alternative weight functions for the frequency and, more generally, wavelet domain. In the wavelet domain, denoising is usually performed by shrinkage of the wavelet coefficients. The idea is to preserve the signal and to discard the noise. In our denoising framework, we consider signal as the outlier and noise as the inlier being estimated. We denoise by subtracting the estimated noise, rather than estimating the denoised signal directly. We take an alternative perspective by considering wavelet shrinkage as robust noise estimation. Figure 4.1 reveals typical shrinkage operators as robust estimators of noise. The most popular wavelet shrinkage method is the mean squared error (MSE) minimizing Wiener filter. However, we can also understand the Wiener filter as a robust noise estimator. If we subtract the

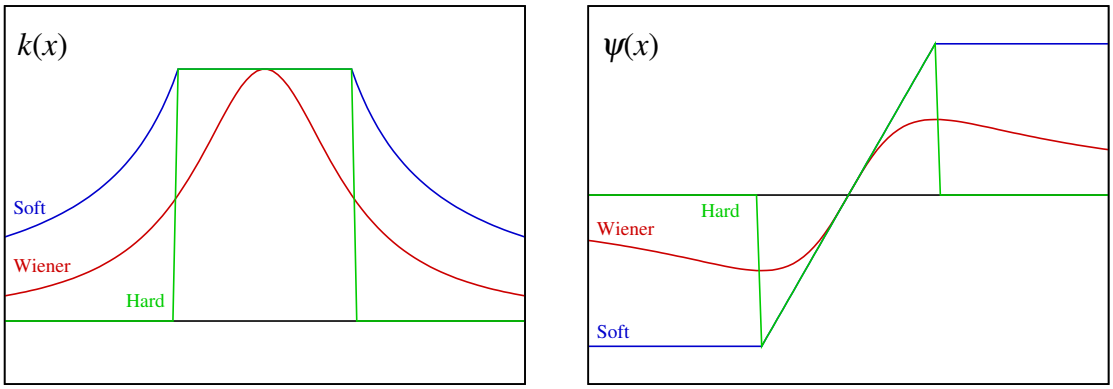


Figure 4.1: *Typical wavelet thresholding and shrinkage functions interpreted as robust estimators for noise. The left plot shows the weight functions $k(x)$ and the right plot shows the corresponding influence functions $\psi(x) = x\,k(x)$. Wiener filtering corresponds to the Lorentzian estimator and has poor outlier rejection. Soft thresholding corresponds to the Huber estimator, which is not redescending. Only hard thresholding has strong outlier rejection.*

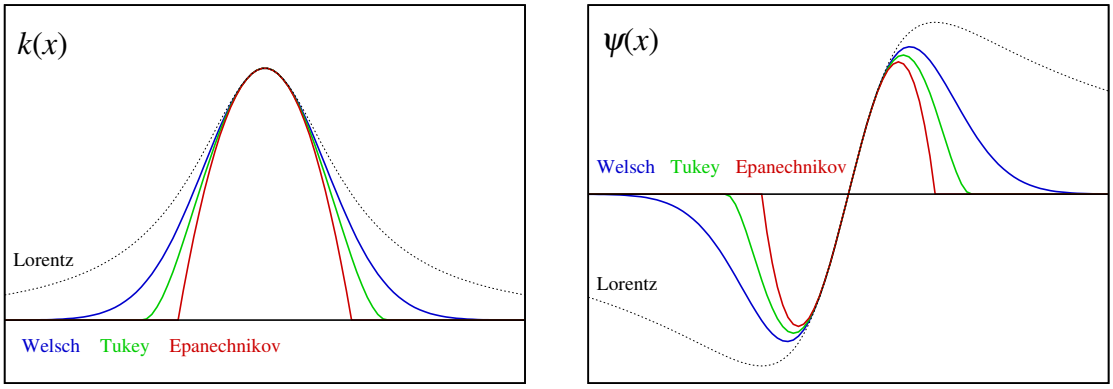


Figure 4.2: *Examples of redescending M-estimators. The left plot shows the weight functions $k(x)$ and the right plot shows the corresponding influence functions $\psi(x) = x\,k(x)$. The Lorentz estimator has the largest tail and does not sufficiently discard outliers. The Welsch estimator has a Gaussian as its weight function. It has a steep descent while still being infinite in support. The weight and influence functions of Tukey and Epanechnikov both have finite supports, rejecting outliers completely. The Wiener filter (Lorentzian estimator) is shown for reference.*

Wiener filter from 1 as

$$K_{p,f} = 1 - \frac{|X_{p,f}|^2}{|X_{p,f}|^2 + \sigma_{p,f}^2} = \frac{1}{|X_{p,f}|^2/\sigma_{p,f}^2 + 1}, \tag{4.13}$$

we recognize the Lorentzian redescending M-estimator. While its influence function is redescending, the Lorentzian has a heavy tail, failing to discard strong signals as outliers, leading to ringing artifacts. The Wiener filter is therefore a poor choice for outlier rejection. The same analysis can be made for thresholding. Soft tresholding is the Huber estimator in disguise, which is not redescending, making it ineffective for robust noise estimation. Hard thresholding, on the other hand, is a redescending M-estimator with strong outlier rejection.

4.5 Outlook

In the following chapters, we show how DDF can be configured using robust noise estimators. By iterating DDF and plugging in dynamically changing robust noise estimators, we will develop image denoising methods that produce state-of-the-art results. Moreover, we will demonstrate that DDF is useful outside of classic image denoising, by using it as a post-processing tool to remove denoising and compression artifacts.

5

Dual-Domain Image Denoising

When the solution is simple,
God is answering.

Albert Einstein

Image denoising methods have been implemented in both spatial and transform domains. Each domain has its advantages and shortcomings, which can be complemented by each other. State-of-the-art methods like (BM3D) therefore combine both domains. However, implementation of such methods is not trivial. We offer a hybrid method that is surprisingly easy to implement and yet rivals BM3D in quality.

The classic image denoising problem is the reconstruction of an image that has been degraded by addition of white Gaussian noise. There are two main classes of image denoising methods: one operates in the spatial domain, the other in a transform domain. The bilateral filter [10] and the non-local means filter [20] are examples of methods which define the filter kernel in the spatial domain. They preserve features like edges, but have difficulties preserving low-contrast details. On the other hand, wavelet thresholding and shrinkage methods operate in a transform domain and excel in preserving details like textures, but suffer from ringing artifacts near edges.

Thus, a hybrid approach is taken by recent works. BM3D [26], *shape-adaptive BM3D* (SA-BM3D) [30], and *BM3D with shape-adaptive principal component analysis* (BM3D-SAPCA) [31], sorted by increasing denoising quality, are considered state-of-the-art. These are sophisticated methods which pay for the high quality with implementation complexity [45]. While producing numerically good results, the methods are not yet perfect [37, 36, 25]. They are based on block matching, which introduces visible artifacts in homogeneous regions, manifesting as low-frequency noise.

We propose a method that is competitive in quality with BM3D, but is much simpler to implement. We combine two popular filters for the two domains using the previously defined DDF. For the spatial domain, we define the bilateral kernel function to be the standard bilateral filter, and for the transform domain, we define it as wavelet shrinkage. The combination of the spatial kernel from the bilateral filter together with the following DFT allow the interpretation of the wavelet transform as a *short-time Fourier transform* (STFT). Furthermore, since the spatial kernel of the bilateral filter is a Gaussian, the wavelet transform is in fact a Gabor wavelet transform.

The bilateral filter is known for its edge-preserving properties. It retains high-contrast features like edges, but cannot preserve low-contrast detail like textures without introducing noise. STFT wavelet shrinkage on the other hand results in good detail preservation, but suffers from ringing artifacts near steep edges. By combininge these two classic methods, we produce a new one which denoises better than when used separately.

5.1 DDF Parametrization

Using the guided DDF, we can compactly describe DDID. We start with $x_0 = y$ and iteratively improve the guide image x_n as

$$x_{n+1} = y - \operatorname{Im} \mathrm{DDF}_n(x_n + iy), \tag{5.1}$$

where $\mathrm{DDF}_n(\cdot)$ returns the noise estimate for iteration n. We define $\mathrm{DDF}_n(\cdot)$ by changing the kernel functions $k_n(\cdot)$ and $K_n(\cdot)$ for every iteration n. We use the following confidence factors and kernel functions

$$a = A = 1 \tag{5.2}$$

$$k_n(d^2, \rho^2) = e^{-\frac{d^2}{\gamma_r(n)\sigma^2}} e^{-\frac{\rho^2}{2\sigma_s^2}} \tag{5.3}$$

$$K_n(D^2) = 1 - e^{-\frac{\gamma_f(n)\sigma^2}{D^2}}. \tag{5.4}$$

Empirically, we found that iterating the filter three times with the following parameters works best:

$$\gamma_r(n) = (100, 8.7, 0.7) \tag{5.5}$$

$$\gamma_f(n) = (8.0, 0.4, 0.8) \tag{5.6}$$

$$\sigma_s = 7. \tag{5.7}$$

The parameter γ_r is the scale factor for the range of the bilateral kernel function. Likewise, the parameter γ_f is the scale factor for the range of the frequency kernel function. The parameter σ_s controls the extent of the spatial Gaussian. We chose the neighborhood kernel radius $r = 15$. Evaluating the bilateral kernel $k_n(\cdot, \cdot)$ function yields the standard bilateral kernel. The frequency range kernel $K_n(\cdot)$ however is a wavelet shrinkage kernel, formulated as a robust noise estimator.

5.2 Discussion

We can gain an intuition about the collaboration of the two domains by denoising a 1D signal. We specify the input as a rectangular function modulated with a sine wave, to which white Gaussian noise was added. Figure 5.1 illustrates the intermediate steps in the spatial and frequency domain for the first two iterations. The first bilateral filter step uses a large range parameter γ_r. This retains the large steps of the rectangle, but smoothens the rest of the signal. The following STFT step recovers the previously lost detail without being affected by the edges of the rectangle. The filtered signal is fed as a guide to the second iteration step, which uses a smaller range parameter. This time, the bilateral filter keeps the recovered edges from the previous STFT step. Although the first STFT step reintroduced ringing artifacts, the bilateral filter recognizes them as noise and filters out. The second STFT step reinforces the detail features in the center but does not bring back the ringing. As observed, the joint bilateral filter has the power to "heal" ringing artifacts caused by the wavelet shrinkage of the preceding iteration; this phenomenon has previously been exploited by Yu et al. [46]. Figure 5.2 shows the evolution of the guide signal x_n over the three iterations. With every iteration, the noise decreases, while only little bias is introduced. Figure 5.3 shows averaged plots over 200 denoised signals. They demonstrate that DDID avoids noisy ringing artifacts typical to STFT with Wiener filter.

Figure 5.4 depicts the denoising progress for a grayscale image. The first row shows the first iteration, with the initial bilateral filtered image on the left. The STFT step yields the details shown in the middle. The sum of the two images give the improved image on the right. This improved image of every iteration is used as an oracle or "guide" for the following iteration. Over the iterations, the bilaterally filtered image becomes sharper, and less details are added, and the noise in the resulting images is reduced.

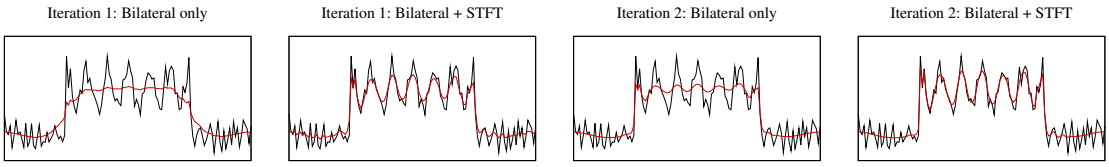


Figure 5.1: *Intermediate steps of first two iterations of denoising. The bilateral filter and* STFT *shrinkage cooperate in alternation to denoise the signal. Denoised results in red are compared against the initial noisy signal in black.*

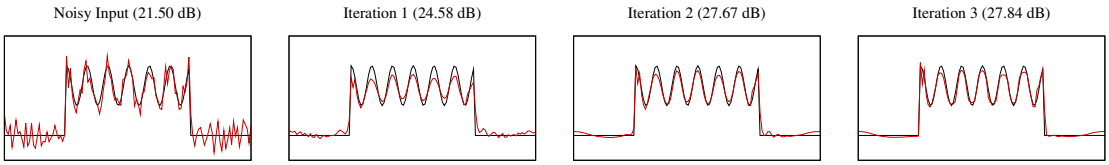


Figure 5.2: *Progression of denoising. The guide signal in red improves every iteration, approximating the original signal in black.*

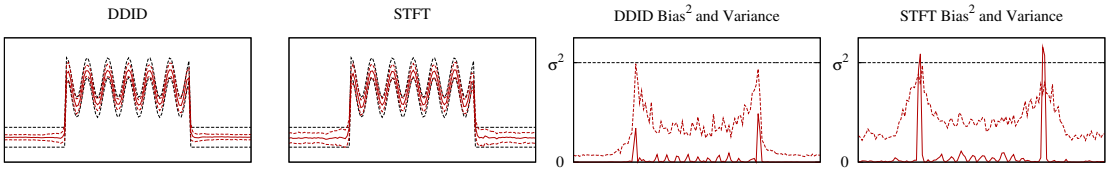


Figure 5.3: *Comparison of* DDID *against* STFT *with Wiener filtering, averaged over 200 signals.* STFT *has residual noise due to ringing artifacts, while* DDID *benefits from using the bilateral mask. Left: red solid and dashed lines denote the expected value with a confidence interval of one standard deviation. Right: red solid and dashed lines are the squared bias and variance respectively. Black dashed lines are the initial noise variance.*

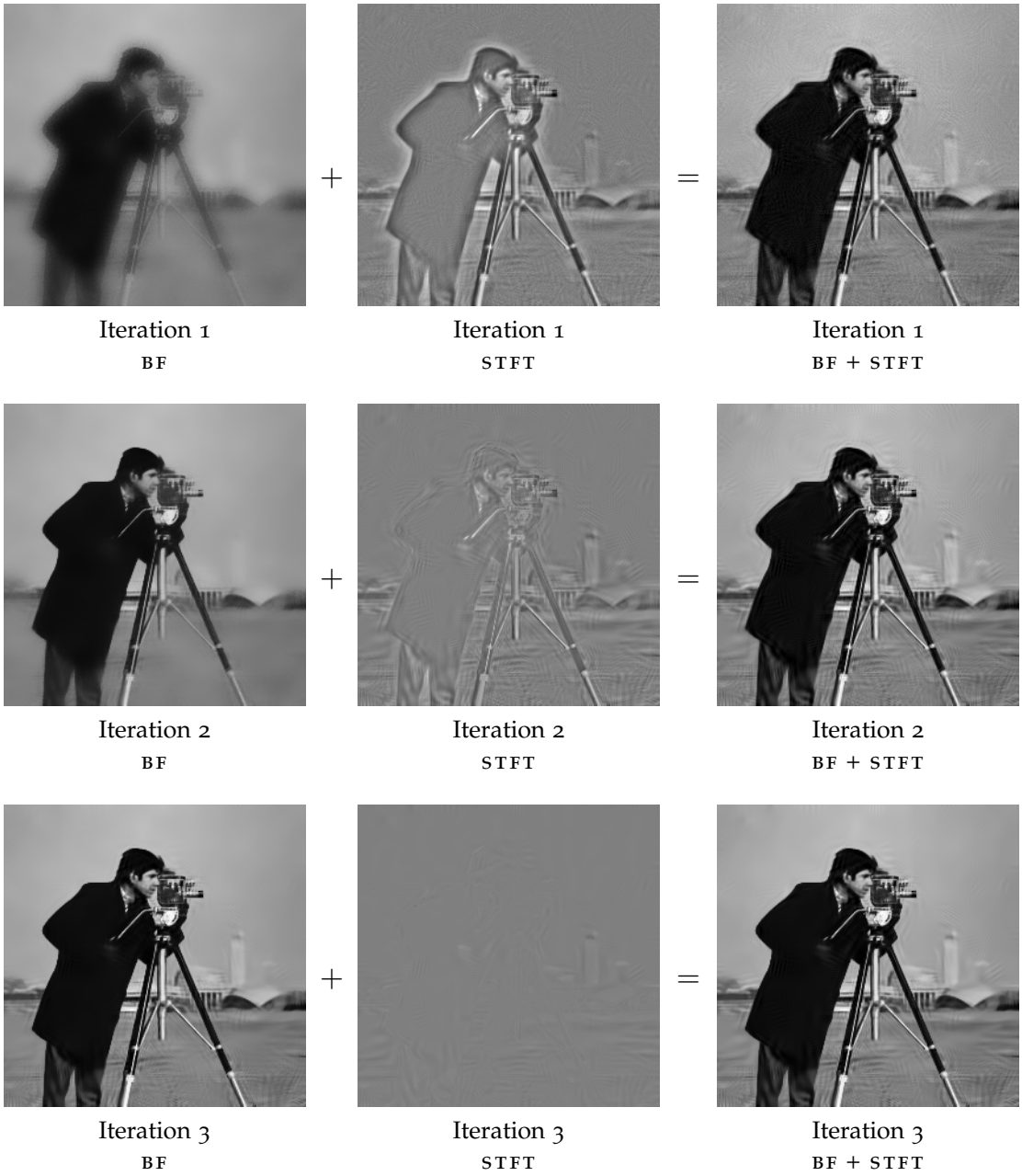


Figure 5.4: *Progression of denoising a grayscale image with* DDID. *Each row consists of an iteration step. In the left columns are the base images from the* BF *step, the center columns contains the detail images from the* STFT *step, and the right column contains the result of the step as a sum of the previous two columns.*

Another way to analyze the denoising process is by observing the evolution of kernels. Unfortunately, we cannot easily express the dual-domain defined kernel in closed form. The denoising actions in the spatial domain and in the frequency domain are time-interleaved and do not occur simultaneously. Therefore, it is unpractical to define a unified dual-domain kernel. However, we can consider the bilateral kernel k_q as an approximation of the dual-domain kernel $\hat{k}_q$. Another approximation of the dual-domain kernel can be made by assuming the original image x is zero mean, allowing us to ignore the bilaterally filtered value. The resulting approximation of the dual-domain kernel is $\hat{k}_q \approx \delta - k_q \mathcal{F}^{-1}\{K_f\}$. Figure 5.6 and Figure 5.7 show the evolution of these kernels involved. The kernels are evaluated at the points marked in Figure 5.5.

Figure 5.5: *Evaluation points for kernel analysis. The first pixel is in the sky, next to the arm of the* Cameraman. *The second pixel is on one of the pillars of a building.*

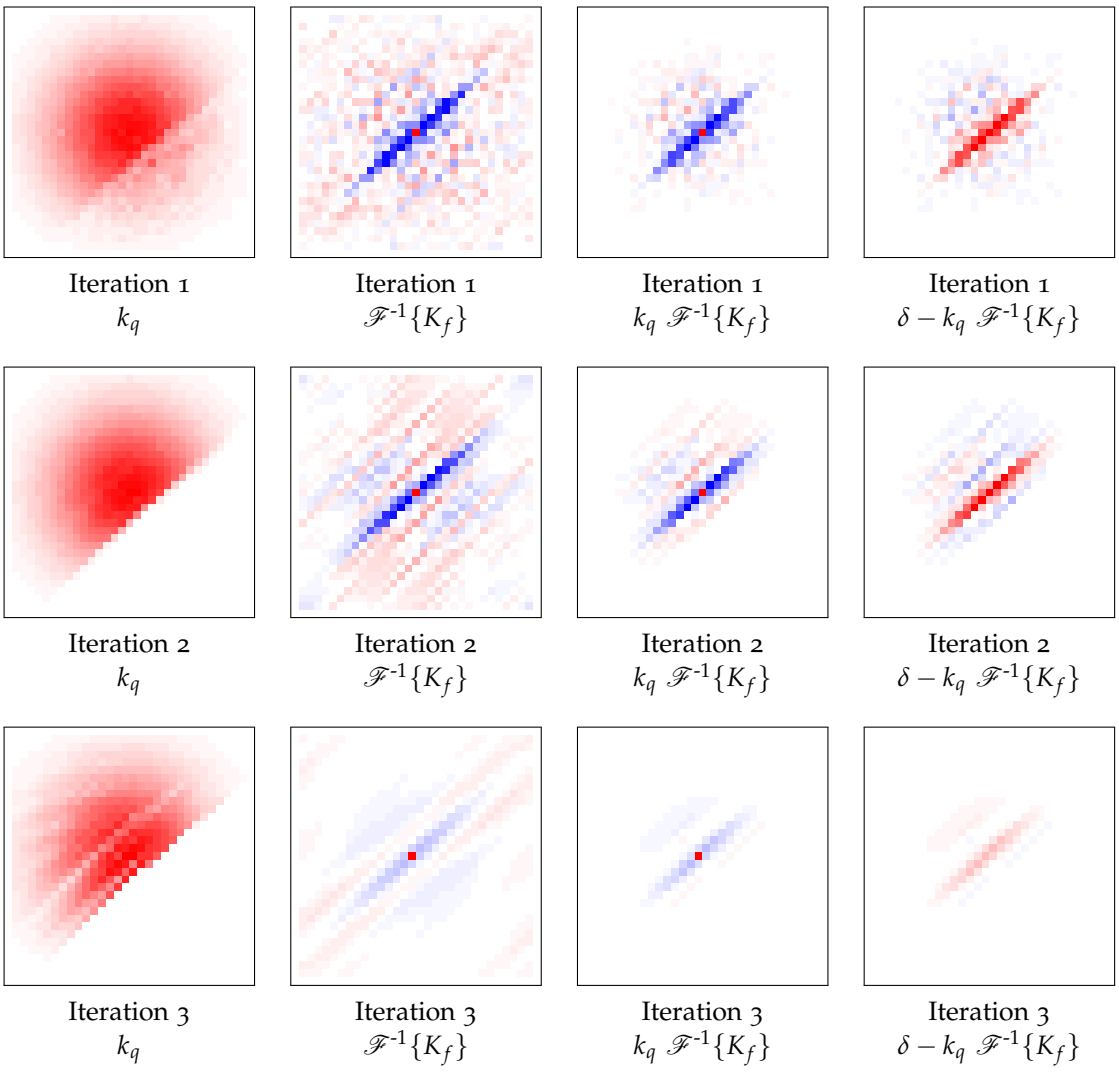


Figure 5.6: *Evolution of kernels near the arm of the* Cameraman. *Red values are positive and blue values are negative. The left most filter is the bilateral kernel. The second kernel is the inverse Fourier transform of the frequency domain kernel, followed by its product with the bilateral kernel, used to estimate the noise. The last kernel is the final kernel if it was directly applied to the noisy image y.*

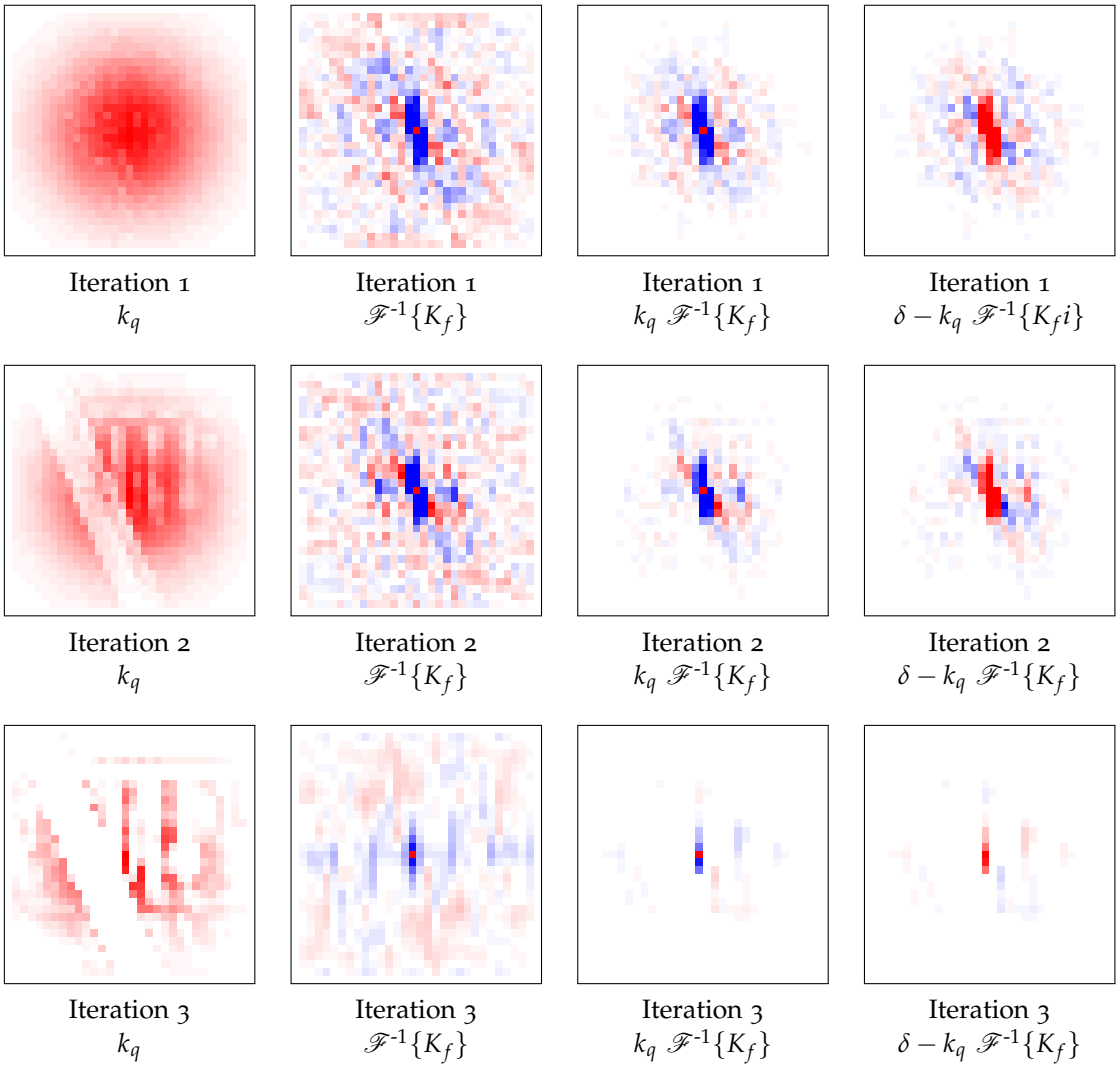


Figure 5.7: *Evolution of kernels for pillar region of the* Cameraman. *The bilateral filter improves over time by eventually discarding the tripod as outlier. In the first two iterations, the inverse Fourier trasnform of the frequency kernel is dominated by the diagonal direction of the tripod. In the last iteration, the tripod is masked out by the bilateral filter and horizontal frequencies dominate. The inverse Fourier transform of the frequency domain kernel is symmetric. the pillars extend to the right and to the left. Masking this kernel with the bilateral filter discards the incorrect pillars on the left. The final kernel contains the original pillar and similar pillars to the right.*

Grayscale	DDID	BM3D	Color	DDID	BM3D
Barbara	**30.80**	30.72	Baboon	**26.17**	25.95
Boats	29.79	**29.91**	F-16	**32.88**	32.78
Cameraman	**29.46**	29.45	House	32.69	**33.03**
Couple	29.56	**29.72**	Kodak 1	29.09	**29.13**
Finger Print	27.32	**27.70**	Kodak 2	32.29	**32.44**
Hill	29.71	**29.85**	Kodak 3	**34.55**	34.54
House	32.66	**32.86**	Kodak 12	33.46	**33.76**
Lena	**32.14**	32.08	Lake	**28.85**	28.68
Man	**29.62**	**29.62**	Lena	**32.30**	32.27
Montage	**32.61**	32.37	Pepper	**31.25**	31.20
Pepper	**30.26**	30.16	Tiffany	**32.49**	32.23

Table 5.1: PSNR (dB) comparison between DDID and BM3D for noise sigma $\sigma = 25$.

5.3 Results

DDID produces competitive results. Table 5.1 compares the PSNR of DDID and BM3D for all the BM3D test images. We chose $\sigma = 25$ (PSNR = 20.17 dB) as the standard deviation of the noise. Numerically, BM3D and DDID show comparable denoising quality.

Figure 5.8 demonstrates that low-frequency noise present in BM3D images is absent in DDID images. Figure 5.9 shows another strength of DDID. The error comparison shows that DDID has smaller errors than BM3D for hair texture. BM3D on the other hand works well for edge-like structures as found in architecture or in the blue ridges in the cheeks of the mandrill.

Figure 5.10 studies noise induced artifacts. Random noise can generate patterns which can be confused as signal. DDID and BM3D-SAPCA retain the wavy patterns in the noise, while BM3D smoothes them away. The latter should be considered a mistake as on other occasions the pattern could indeed have been a valid signal. Recall that our method is much simpler to implement than BM3D and especially BM3D-SAPCA, which extends BM3D by additional shape-adaptive and PCA steps.

With an optimized MATLAB implementation of DDID, denosing a grayscale image with 256×256 pixels and a window size of 31×31 using a single core of a Intel Xeon 2.67 GHz takes 70 seconds. The bottleneck is the transition from spatial to frequency domain. If this transition was a pure Gabor transform, we could exploit sliding window techniques [47] to update the Fourier coefficients incrementally. However, since the signal is multiplied by an arbitrary range kernel, we need a per-pixel FFT with complexity $O(N^2 \log N)$. Thus, we implemented a C version using the FFTW library, which shortened the time

to 40 seconds. Since the pixels are mutually independent, we achieved linear scalability using dual quad-core CPUs, reducing the time to 5 seconds. Finally, our GPU implementation on an NVIDIA GeForce GTX 470 cut the time down to one second.

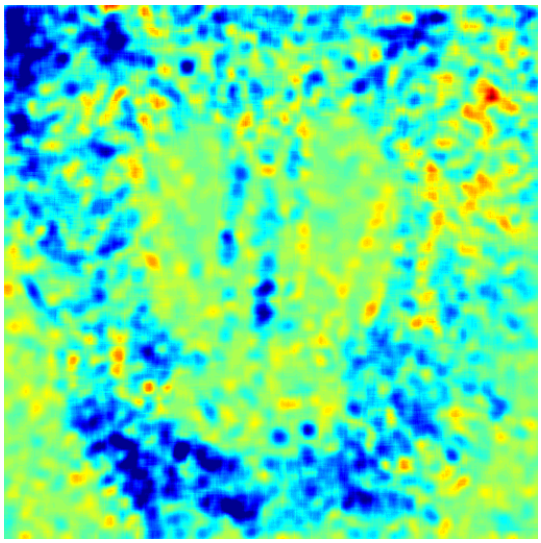
5.4 Conclusions

In this chapter, we have presented DDID, a simple image denoising method which works surprisingly well. Its quality is comparable to BM3D, while being much easier to implement. The simplicity and effectiveness comes from the DDF it is based on. This first result with DDF invites for more experimentation to achieve even higher quality.

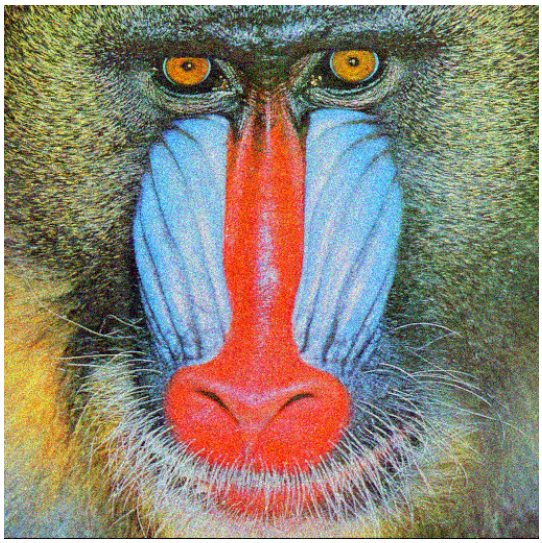
Original | Noisy Image (20.17 dB)

BM3D (32.37 dB) | **DDID (32.61 dB)**

Figure 5.8: Comparison of DDID and BM3D for denoising a grayscale image. DDID has less low-frequency noise than BM3D.



Error Difference



Noisy Image (20.17 dB)

BM3D (25.95 dB)

DDID (26.17 dB)

Figure 5.9: Comparison of DDID and BM3D. DDID effectively denoises hair-like structures, while BM3D is stronger for edge-like structures. Blue regions mark where DDID has lower error than BM3D, yellow and red regions mark the opposite, and green regions mark similar errors.

Original | Noisy Image 20.17 dB | BM3D 30.16 dB | DDID 30.26 dB | **BM3D-SAPCA 30.43 dB**

Figure 5.10: Artifact comparison of DDID, BM3D, and BM3D-SAPCA. Misclassification of noise as signal due to accidently regular patterns is acceptable and is to be expected. BM3D produces a smooth result, but this would fail in other occasions where the pattern would be a signal.

6

Progressive Image Denoising

When I am working on a problem, I never think about beauty
but when I have finished, if the solution is not beautiful,
I know it is wrong.

Buckminster Fuller

In the last chapter we have shown with DDID that by simply iterating DDF three times, we can denoise images effectively. While the results of DDID are qualitatively comparable to BM3D, both methods still have artifacts. Both DDID and BM3D produce acceptable images for natural images, but human vision is less forgiving when viewing synthetic images. Objective metrics have difficulties capturing the nuances our eyes are sensitive to. The numerically best image denoising method is BM3D-SAPCA, as shown in Table 6.1. BM3D-SAPCA achieves exceptional PSNR values, but still suffers from visible artifacts, as can be seen in Figure 6.1.

In this chapter, we propose *progressive image denoising* (PID), a method that produces minimal artifacts for both natural and synthetic images. Compared to DDID, this method takes iteration to the extreme. Moreover, our new formulation of image denoising gives new insights into the subject by connecting high-quality image denoising to statistical mechanics.

PID is a method based on *deterministic annealing* (DA) and robust noise estimation. DA is a heuristic method that is efficient at solving complex optimization problems where many local extrema exist. We combine DA with redescending M-estimators, similar to previous work of Li [48] and Frühwirth and Waltenberger [49]. We use the DDF framework to define this redescending M-estimator. Our method produces high-quality results, void of artifacts typical to patch-based methods. It performs not only well for natural images, but also for synthetic images where artifacts are usually more apparent.

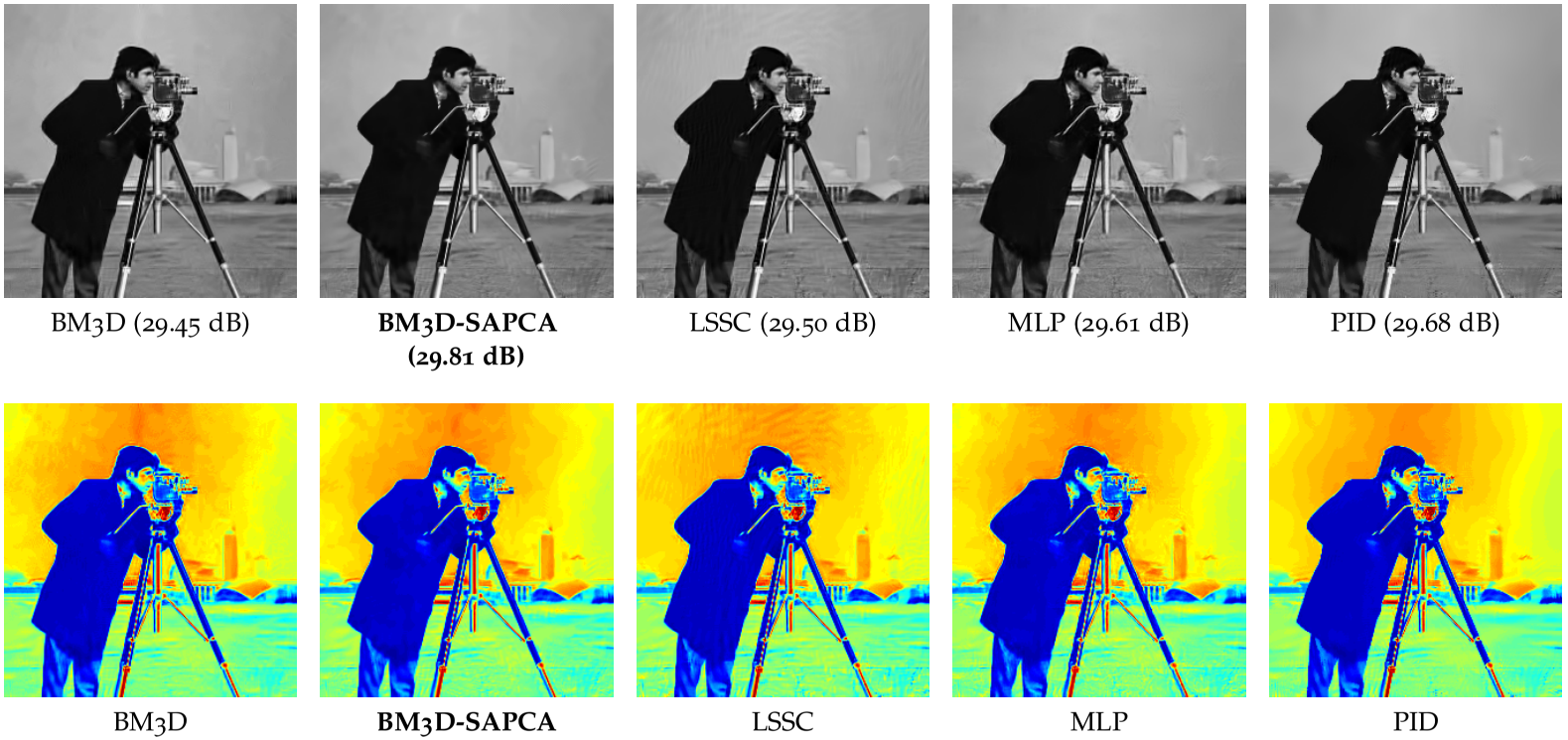


Figure 6.1: *Comparison of denoising* Cameraman *with noise sigma* $\sigma = 25$*. While the* PSNR *values of the considered methods are all high, only* PID *manages to produce homogeneous sky regions without objectionable artifacts. The bottom row shows false color images to highlight the artifacts in the sky.*

We describe the method in three steps. In Section 6.1, we first cast image denoising as a gradient descent problem and identify the gradient with a noise differential. Then, in Section 6.2, we estimate the noise differential using redescending M-estimators in the spatial and frequency domains. To complete the algorithm, we define the annealing schedule in Section 6.3. The parameters of the algorithm are given in Section 6.4. In Section 6.5, we present high-quality results and compare them to state-of-the-art methods and conclude in Section 6.6.

6.1 Denoising as Gradient Descent

We start with the original "ill-posed" problem formulation given by Equation 1.1,

$$y = x + n. \tag{6.1}$$

In practice, we can only *estimate* a decomposition $y = \tilde{x} + \tilde{n}$. Formulated as an optimization problem, we look for $\tilde{x}$ that minimizes the apriori unknown energy $E = (\tilde{x} - x)^2$. However, if we had an estimate of this energy, we could derive a gradient as $\partial\tilde{E}/\partial\tilde{x} = 2(\tilde{x} - x)$, which is the noise of $\tilde{x}$, up to scaling.

Starting with $x_0 = y$, we can perform a gradient descent, where the gradient is the momentarily estimated noise differential n_i for the current estimate x_i. The noise differential n_i integrates over time i to the estimated total noise instance as $\tilde{n} = \sum_{i=0}^{\infty} n_i$. The gradient descent formulation is thus

$$x_{i+1} = x_i - n_i. \tag{6.2}$$

6.2 Robust Noise Differential Estimation

In Chapter 5, we used DDF to estimate the total noise in every iteration using Equation 5.1. Here, we use DDF to estimate the noise *differential* n_i. Equation 6.2 becomes

$$x_{i+1} = x_i - \operatorname{Im} \mathrm{DDF}_i(x_i + iy) \tag{6.3}$$

Instead of estimating the noise anew in every iteration, the noise is progressively removed, hence the name for our method. We empirically found that progressive removal of noise is possible by specifying the following. First, we perform unguided DDF, which is equivalent to identifying the noisy image y with the current guide image x_i as $y = x_i$. Second, we discard the noise $\bar{n}$ estimated by

the bilateral filter, which is equivalent to setting the spatial confidence factor $a = 0$ in Equation 4.3. Third, we specify the gradient descent step by the frequency domain confidence factor A of Equation 4.8.

6.3 Shape Shifting Estimator

The missing ingredient to our method is the dynamic parameterization of the robust noise estimator. Time-varying robust estimators have been used by Li et al. [48] to replace scale selection by deterministic annealing. Frühwirth and Waltenberger [49] allowed other kernel shape parameters to change over time as well. Similarly, we adapt the annealing concept in a flexible style. While we shrink the scale of the range kernel k_r like in traditional DA, we simultaneously enlarge the spatial kernel k_s over time by defining

$$k_i(d^2, \rho^2) = k_r\left(\frac{d^2}{T_i}\right) k_s\left(\frac{\rho^2}{S_i}\right). \tag{6.4}$$

The resulting dynamic bilateral kernel $k_i(\cdot)$ is schematically depicted in Figure 6.2. Our frequency range kernel function is independent of time and we define it as

$$K_i(D^2) = K(D^2). \tag{6.5}$$

We define the scale parameters T_i and S_i for the range and spatial kernel of the bilateral kernel function $k_i k$ as functions of time i:

$$T_i = \sigma^2 \gamma_r \alpha^{-i} \tag{6.6}$$

$$S_i = \sigma_s^2 \gamma_s \alpha^{i/2} \tag{6.7}$$

The first scale parameter T_i of the range kernel k_r is our temperature which is reduced over time. We found that an exponential decay of the temperature works best, where α^{-1} is the rate of this decay. γ_r is a large initial scale factor. The second scale parameter S_i, however, we let grow. When the temperature T_i is high, the range kernel k_r covers the entire dynamic range and the spatial kernel k_s should be small to reduce bias from neighbouring pixels in the noise estimation. On the other hand, when the temperature is low, the range kernel becomes narrow and we require larger spatial support to discern autocorrelated signal with small amplitudes from noise. When the temperature has totally cooled down, the range kernel is a Dirac delta, and the spatial kernel is the constant 1, covering the entire spatial domain. Similar to the parameter γ_r of

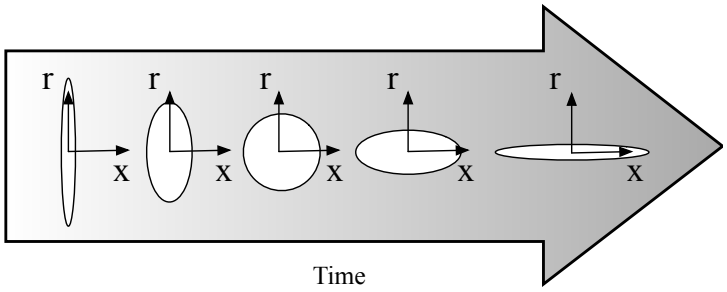


Figure 6.2: *Evolution of the bilateral kernel $k_r k_s$. Deterministic annealing changes the scale parameters by shrinking T_i and increasing S_i over time. The axis 'r' and 'x' represent the range and spatial dimensions.*

the range kernel, γ_s is a small initial scale factor for the scale σ_s of the spatial kernel. The frequency domain kernel function $K_i(\cdot)$ we specify simply as a robust kernel $K(\cdot)$, independent of time i.

6.4 Implementation

Many denoising methods split their algorithms into two steps. First, an oracle is computed, and then the oracle is used to denoise the noisy image. The solution of deterministic annealing is only near-optimal as we may still end up in a local minimum. We therefore follow the same pattern like other methods and perform an additional denoising step. We discovered that the resulting image from the annealing stage serves as an excellent oracle to denoise the noisy signal with a single DDID step. We use Gaussians for all the kernels of the robust noise estimator, i.e., $k_r(d^2) = k_s(d^2) = K(d^2) = e^{-d^2}$. In Section 6.5.2, we experiment with robust estimators that have stronger outlier rejection.

The parameters in the algorithm were empirically found. We use $N = 30$ iterations with a temperature decay rate of $\alpha^{-1} = 1.533^{-1}$ and gradient step size $A = 0.567 \log \alpha$. These parameters change together. For example to double the number of steps, we would perform $N \to 2N$, $\alpha \to \sqrt{\alpha}$, and $A \to A/2$. The initial scale factor for the range scale is $\gamma_r = 988.5$, and for the spatial scale $\gamma_s = 2/9$. The window radius is $r = 15$, and we use a reference spatial sigma of $\sigma_s = 7$. For the final DDID step, we use a larger kernel size with window radius $r = 31$ and spatial sigma $\sigma_s = 16$. The range and frequency domain parameters are $\gamma_r = 0.6$ and $\gamma_s = 2.16$.

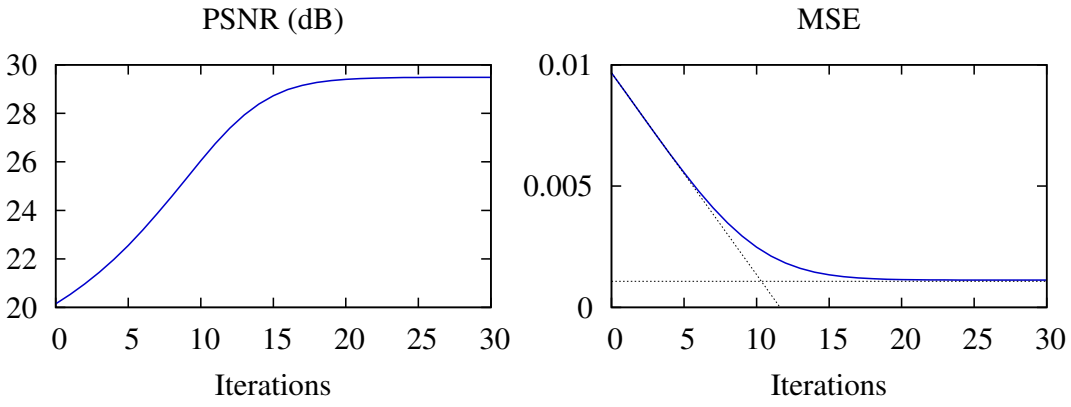


Figure 6.3: PSNR *and* MSE *improvement over time. The* PSNR *increases nearly linearly for most of the time. The* MSE *follows a hyperbolic curve with the bias as one of its asymptotes.*

6.5 Results

We present the results of our algorithm. Figure 6.5 displays the denoising process as an evolution starting with the noisy image. We used 30 iterations, where the intermediate images are snapshots taken after 10, 20, and 30 iterations. Usually, a denoising output of an iteration step cannot be used as input for another step, as the output pixels are correlated and estimating the variance would require expensive covariance tracking. In our case, however, we remove correlated noise in the spatial domain that have wave-like struture. In the frequency domain, these waves are decorrelated and therefore, no covariance tracking is needed. This allows the iteration to go forward without getting stuck in a local minimum.

We analyze the denoising process for the familiar *cameraman*. Figure 6.3 shows the improvement of quality over time. The PSNR increases fast in the beginning, and slows down as the noise becomes smaller. The corresponding MSE exhibits a strong linear decrease in the beginning and asymptotically approaching the squared bias as the variance vanishes. The plots look similar for any image we denoised.

We gain a better understanding about the annealing process by observing the evolution of pixels marked in Figure 6.4. In Figure 6.6, we consider two pixels, x_q on the tall building and x_p in the sky. The difference is relatively small, in the range of the noise sigma, since $\sigma = 25/255 \approx 0.1$. In the first plot, we see the noise differentials of $n_{i,q}$ and $n_{i,p}$ as a function of time. Since the noise is uncorrelated, the noise differentials also take "independent" paths. They first rise steeply and then fall off, eventually vanishing. The second plot shows the

Figure 6.4: *Red and blue pixels mark the locations of measurement x_p and x_q for analyzing the annealing process.*

integrals of the noise differentials, which are the range trajectories $x_{i,q}$ and $x_{i,p}$ of the pixels, starting with the noisy values y_q and y_p. We observe a smooth fall off towards the vicinity of their ground truth values, suggested by dashed lines. The third plot shows the relation between the two pixels as the squared difference/gradient $(x_{i,q} - x_{i,p})^2$ between them. For the first 15 iterations, this squared difference is smaller than the temperature T, so the squared gradient is considered as noise from the perspective of the spatial range kernel k_r. This means that the two pixels affect each others noise estimates. After the two curves cross each other, the temperature is below the squared gradient and the interaction between the two pixels are "frozen". This can be better seen in the last plot: the normalized Euclidean distance, i.e., the quotient $\frac{(x_{i,q}-x_{i,p})^2}{T_i}$ becomes larger than 1 after iteration 15, which in turn lets the weight of the range kernel $e^{-\frac{(x_{i,q}-x_{i,p})^2}{T_i}}$ vanish quickly. After iteration 20, the squared gradient between the pixels are considered outliers for both pixels.

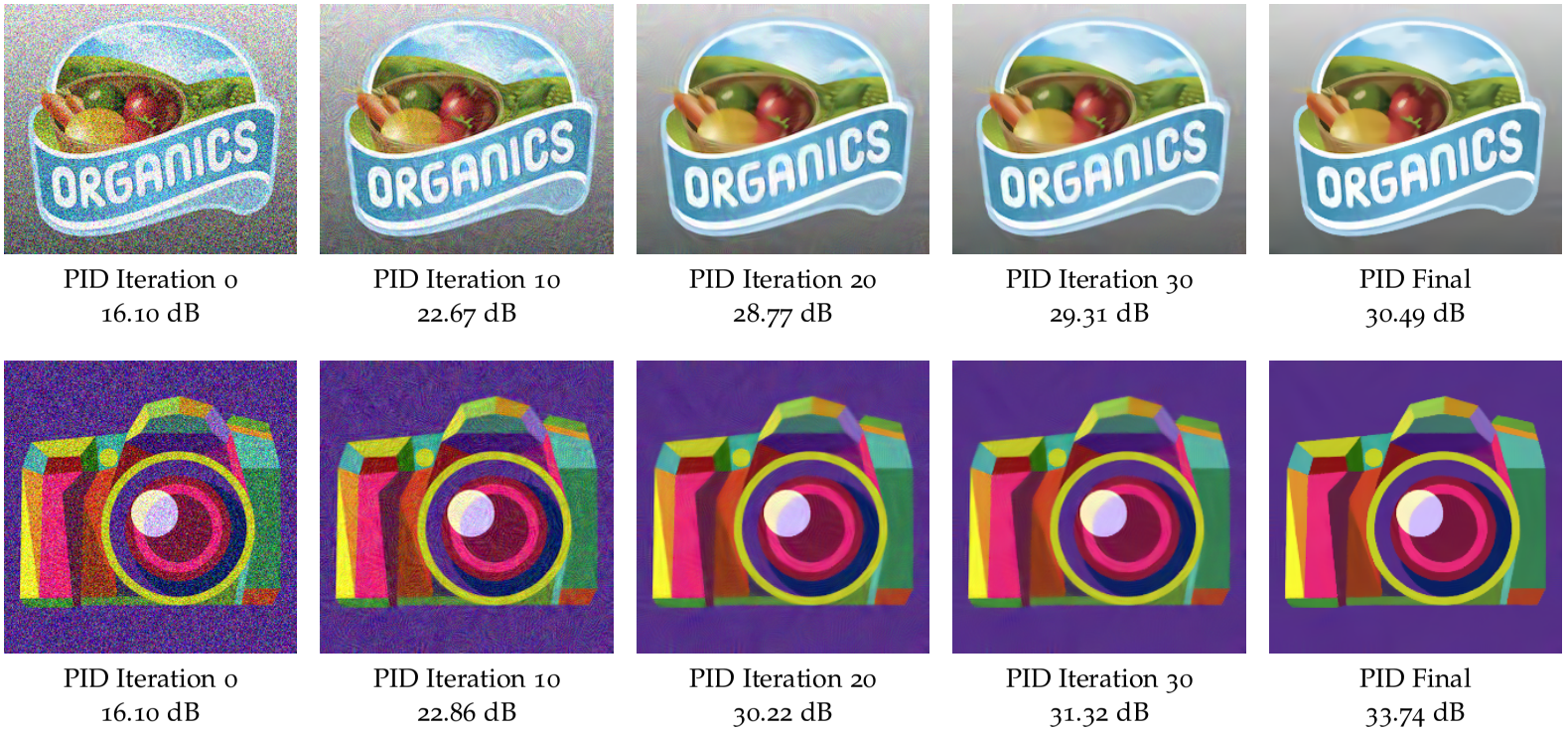


Figure 6.5: *Denoising progress of* PID. *The noise sigma is* $\sigma = 40$. *The image* PID Iteration 30 *is used as an oracle to produce the final image with a single* DDID *step.*

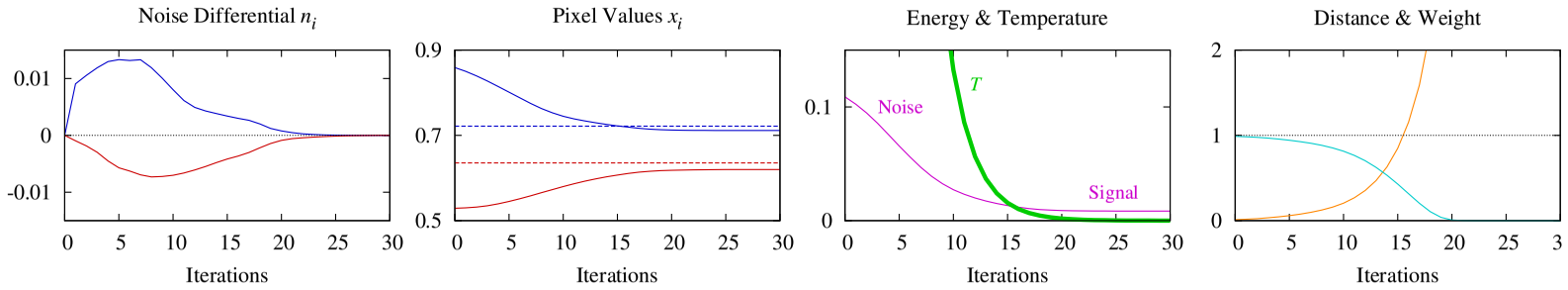


Figure 6.6: *Study of the evolution of two pixels across an edge on* cameraman *(Figure 6.1). Pixel q (blue) is on the tall building, pixel p (red) is in the sky, to the right of the building. The left most plot shows the evolution of the noise differentials $n_{i,q}$ and $n_{i,p}$ of the two pixels. They take individual paths and eventually converge to 0 when no change happens to either pixel. The second plot displays the pixel values $x_{i,q}$ and $x_{i,p}$, which are the time integrations of the noise differentials. The dashed lines are the ground truth values. The third plot displays the energy (purple), i.e., the squared difference $(x_{i,q} - x_{i,p})^2$ between the two pixels and the faster decreasing temperature T_i (green). The curves cross near iteration 15 at which point the interaction between the two pixels slows down. Up to this point the pixel difference was considered noise, after this point the pixel difference is considered signal. Note that the energy follows a similar curve as the* MSE *in Figure 10.9. The last plot shows the quotient of the previous two curves, i.e., normalized Euclidean distance $\frac{(x_{i,q}-x_{i,p})^2}{T_i}$ (orange), and the Gaussian weight $e^{-\frac{(x_{i,q}-x_{i,p})^2}{T_i}}$ (cyan). After the 20th iteration, there is no more diffusion across the edge.*

6.5.1 Natural and Synthetic Images

Visually, PID produces aesthetically pleasing results. Figure 6.7 demonstrates that our method works well for natural images. Figure 6.8 constrasts the qualities between BM3D and our method for synthetic images. With PID, gradients are smoother, edges are sharper, tips are clearer, and text is crisper. The color coded error image displays the difference of the squared errors of BM3D and PID. Our method has lower error than BM3D for most pixels.

Numerically, PID is even more competitive than DDID. Table 6.1 summarizes results for grayscale images. The grayscale images are the same set tested by BM3D and mostly originate from the *Signal and Image Processing Institute* (SIPI) database. Some of the SIPI/BM3D images have synthetic character with large amount of homogenous regions like sky or gradients. These are *Cameraman, House, Lena, Montage*, and *Pepper*. For these images, we observe most of the gain using PID. Other images have more "natural" character and there is not much to improve. Highly stochastic images like *Finger Print* are hard to improve. Since PID excels at denoising highly correlated images, it is not a surprise that it performs well on color images. Since color channels are highly correlated, signals and noise are better separated. The results for color images are listed in Table 6.2. For most images, especially for high-noise scenarios, PID exceeds BM3D and DDID.

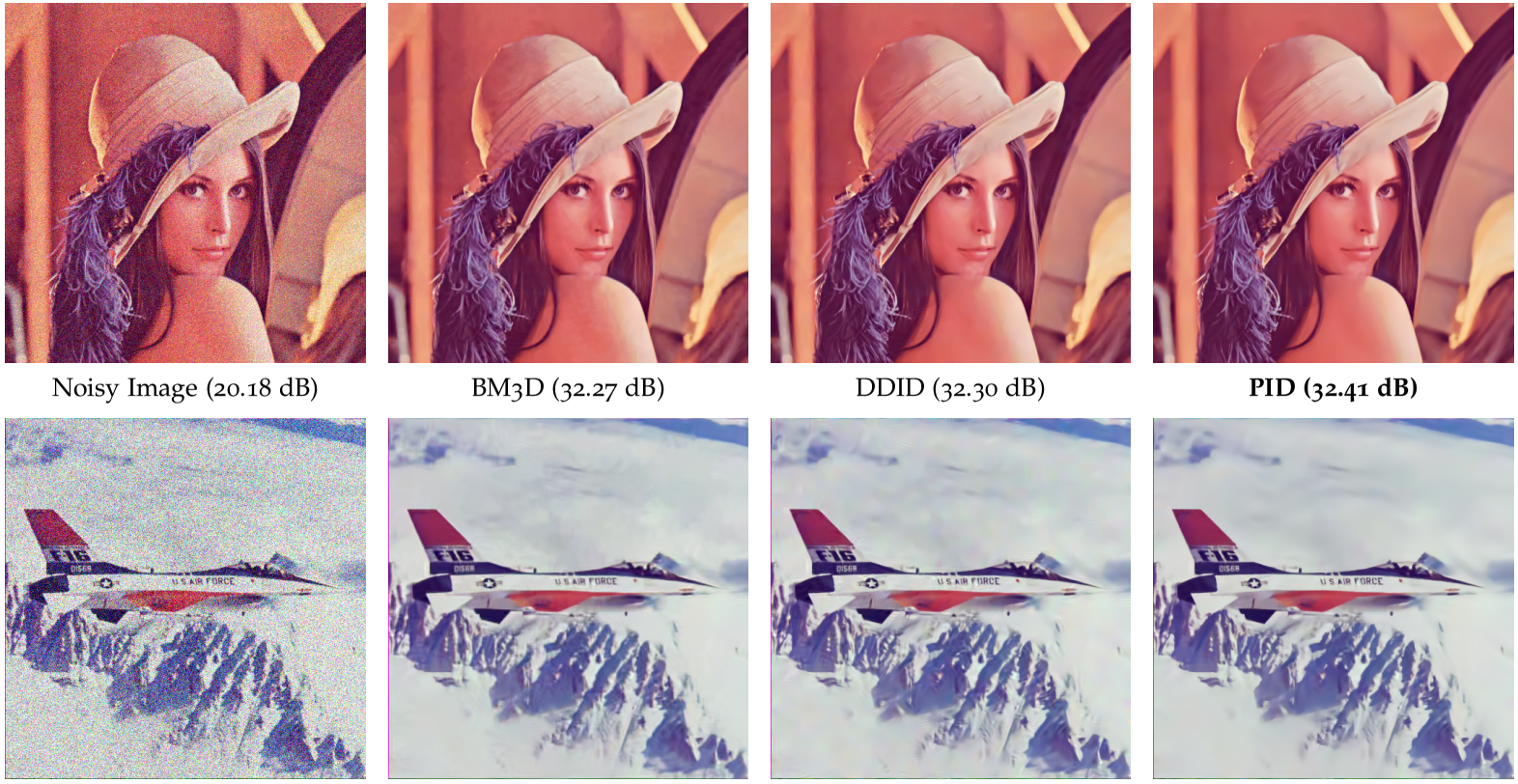


Figure 6.7: *Comparison of denoising natural images. The noise sigma are $\sigma = 25$ and $\sigma = 50$ respectively. Our method produces smooth results for skin and homogeneous regions.*

Figure 6.8: *Comparison of denoising synthetic images with noise sigma $\sigma = 40$. Our method reconstructs cleaner contours and gradients than* BM3D *and* DDID*. The bottom right image shows the difference of the squared errors of* BM3D *and* PID*. Red and yellow regions mark where* BM3D *has a smaller error than* PID*, blue regions mark the opposite, and green regions mark similar errors.*

Grayscale	BM3D	SAPCA	LSSC	MLP	DDID	PID
Barbara	30.72	**31.00**	30.49	29.55	30.80	30.56
Boats	29.91	**30.03**	29.90	29.97	29.79	29.80
Cameraman	29.45	**29.81**	29.50	29.61	29.46	29.68
Couple	29.72	**29.82**	29.67	29.74	29.56	29.64
Finger Print	27.70	**27.81**	27.62	27.65	27.32	27.17
Hill	29.85	**29.96**	29.83	29.88	29.71	29.77
House	32.86	**32.96**	33.13	32.57	32.66	32.84
Lena	32.08	**32.23**	31.86	32.26	32.14	32.12
Man	29.62	29.81	29.70	**29.89**	29.62	29.68
Montage	32.37	**32.97**	32.25	32.03	32.61	32.76
Pepper	30.16	**30.43**	30.23	30.30	30.26	30.34

Table 6.1: *For grayscale images,* BM3D-SAPCA *has the best numerical performance. However, there are several methods which challenge* BM3D*, including* LSSC *[34],* MLP *[35],* DDID*, and* PID*. The noise sigma is* $\sigma = 25$*.*

Color	BM3D	DDID	PID	BM3D	DDID	PID
Baboon	25.95	**26.17**	26.12	23.15	23.31	**23.41**
F-16	32.78	32.88	**33.02**	29.79	29.78	**30.10**
House	**33.03**	32.69	32.90	30.47	29.99	**30.54**
Kodak 1	29.13	29.09	**29.18**	25.86	25.75	25.86
Kodak 2	**32.44**	32.29	32.40	**29.84**	29.61	29.81
Kodak 3	34.54	34.55	**34.70**	31.34	31.09	**31.39**
Kodak 12	**33.76**	33.46	33.55	**30.98**	30.53	30.82
Lake	28.68	28.85	**28.93**	26.28	26.41	**26.57**
Lena	32.27	32.30	**32.41**	29.88	29.82	**30.07**
Pepper	31.20	31.25	**31.36**	28.93	29.13	**29.34**
Tiffany	32.23	32.49	**32.61**	29.83	29.85	**30.12**
		$\sigma = 25$			$\sigma = 50$	

Table 6.2: PSNR *(dB) comparison between* BM3D*,* DDID*, and* PID *for color images with noise sigma* $\sigma = 25$ *and* $\sigma = 50$*. Our method works well for strongly noise contaminated images.*

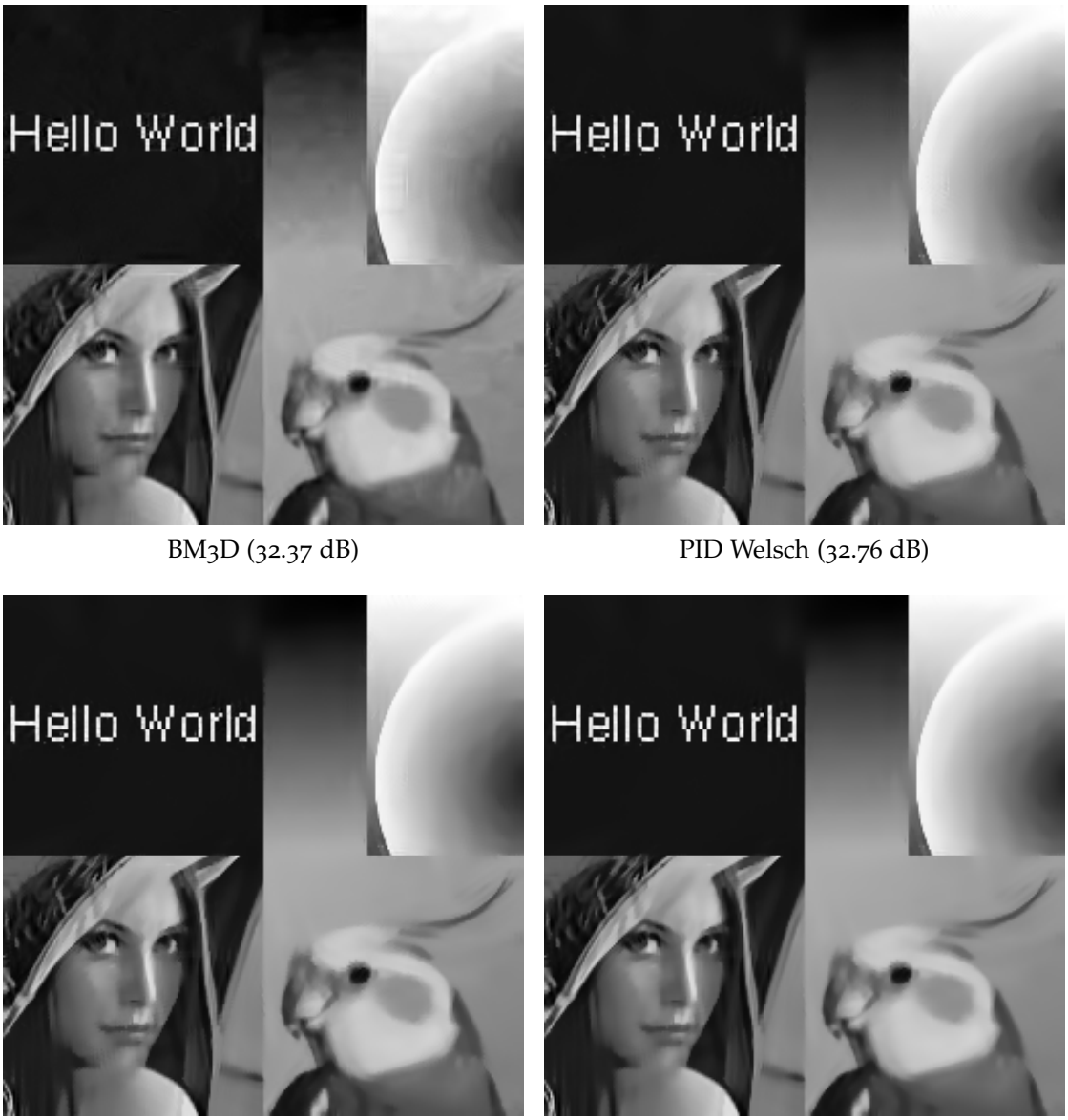
GRAYSCALE	BM3D	DDID	PID^{W}	PID^{T}	PID^{E}
Barbara	30.72	**30.80**	30.56	30.70	30.79
Boats	**29.91**	29.79	**29.80**	**29.81**	29.78
Cameraman	29.45	29.46	**29.68**	**29.64**	**29.55**
Couple	**29.72**	29.56	29.64	29.66	29.62
Finger Print	**27.70**	27.32	27.17	27.25	27.31
Hill	**29.85**	29.71	29.77	29.75	29.68
House	32.86	32.66	**32.84**	**32.90**	**32.95**
Lena	32.08	32.14	32.12	**32.17**	**32.18**
Man	29.62	29.62	**29.68**	**29.66**	29.58
Montage	32.37	32.61	**32.76**	**32.86**	**32.93**
Pepper	30.16	30.26	**30.34**	**30.39**	**30.40**

^{W}Welsch ^{T}Tukey ^{E}Epanechnikov

Table 6.3: PSNR *(dB) comparison of robust estimation kernels for grayscale images with noise sigma* $\sigma = 25$. *Welsch is the most conservative denoiser. Epanechnikov on the other hand is the most aggressive denoiser for images with homogeneous regions like* House, Lena, Montage, *and* Pepper. *Tukey is a compromise between Welsch and Epanechnikov.*

6.5.2 Alternative Robust Estimators

To make use of our flexible DDF framework, we can experiment with robust estimators which have stronger outlier rejection than the Gaussian. The most typical redescending M-estimators besides the Gaussian/Welsch estimator are the Tukey and Epanechnikov estimators, depicted in Figure 4.2. In Table 6.3, we compare the PSNRS using the different robust kernels for the range kernel k_r. All other kernels are Gaussians. A visual comparison is made in Figure 6.9. Smooth images like *House*, *Montage*, and *Pepper* benefit most from using an estimator with strong outlier rejection.



BM3D (32.37 dB) PID Welsch (32.76 dB)

PID Tukey (32.86 dB) **PID Epanechnikov (32.93 dB)**

Figure 6.9: *Comparison of denoising* montage *image using different redescending M-estimators for the range kernel. The noise sigma is* $\sigma = 25$*. Redescending M-estimators with hard outlier rejection like Tukey and Epanechnikov produce numerically better results, but visually, the images look similar.*

6.5.3 Artifacts Study

Artifacts are sometimes hard to spot. When looking at natural images, artifacts may actually increase the perceived realism, due to structures in the image that are recognized as detail. Examples are clouds and surface textures like in Figure 6.7 and Figure 6.10. However, careful observation shows that such structures are not existent in the original image and are in fact residual noise. In synthetic images as shown in Figure 6.8, these artifacts are more apparent since we have an idealized reference image in mind.

Since most image denoising methods are biased towards natural images, where the PSNR results are close to the theoretical limit, it is desirable to have synthetic images which can be evaluated numerically and visually. We found that the Campbell-Robson *contrast sensitivity function* (CSF) charts are suitable candidates. CSF charts are synthetic images used for evaluating human perception of contrast as a function of frequencies. They contain therefore a continuous range of amplitudes and frequencies in a single image. Figure 6.11 and Figure 6.12 demonstrate the effectiveness of CSF charts for evaluating image denoising methods. The table shows that most denoising methods have difficulties in homogeneous regions. Also, the denoised CSF charts reveal characteristic artifacts for every method. The CSF images were generated using the formulas in Section 10.2.2.

Original

Noisy Image (14.16 dB)

BM3D (28.93 dB)

PID (29.34 dB)

Figure 6.10: *The* BM3D *result looks apparently more real and closer to the original, but close inspection reveals details as residual noise. The noise sigma is* $\sigma = 50$.

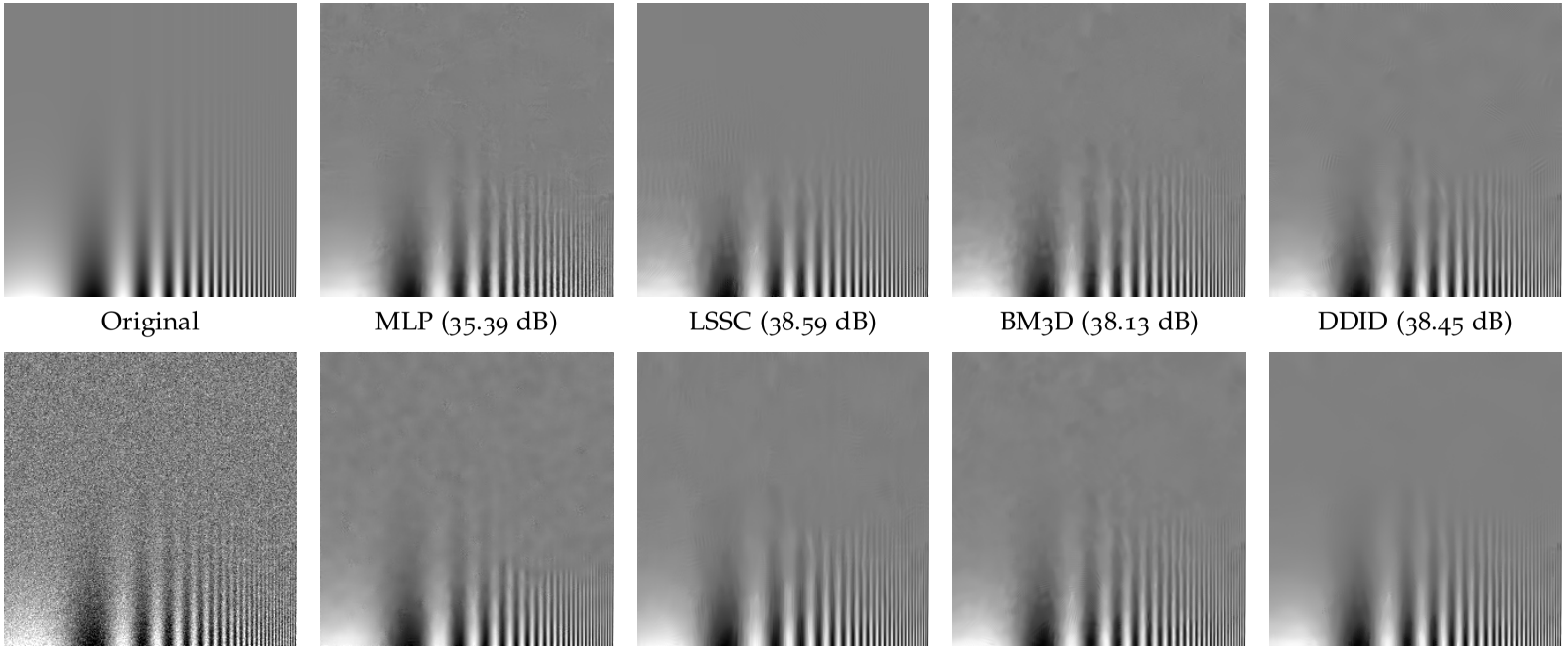


Figure 6.11: *Comparison of denoising* CSF *charts with sinusoidal waves. Noise sigma is* $\sigma = 25$*. This table exposes the characteristic artifacts of each method. Most methods suffer either from loss of contrast or low-frequency noise in homogeneous regions. The results of* PID *are nearly artifact-free.*

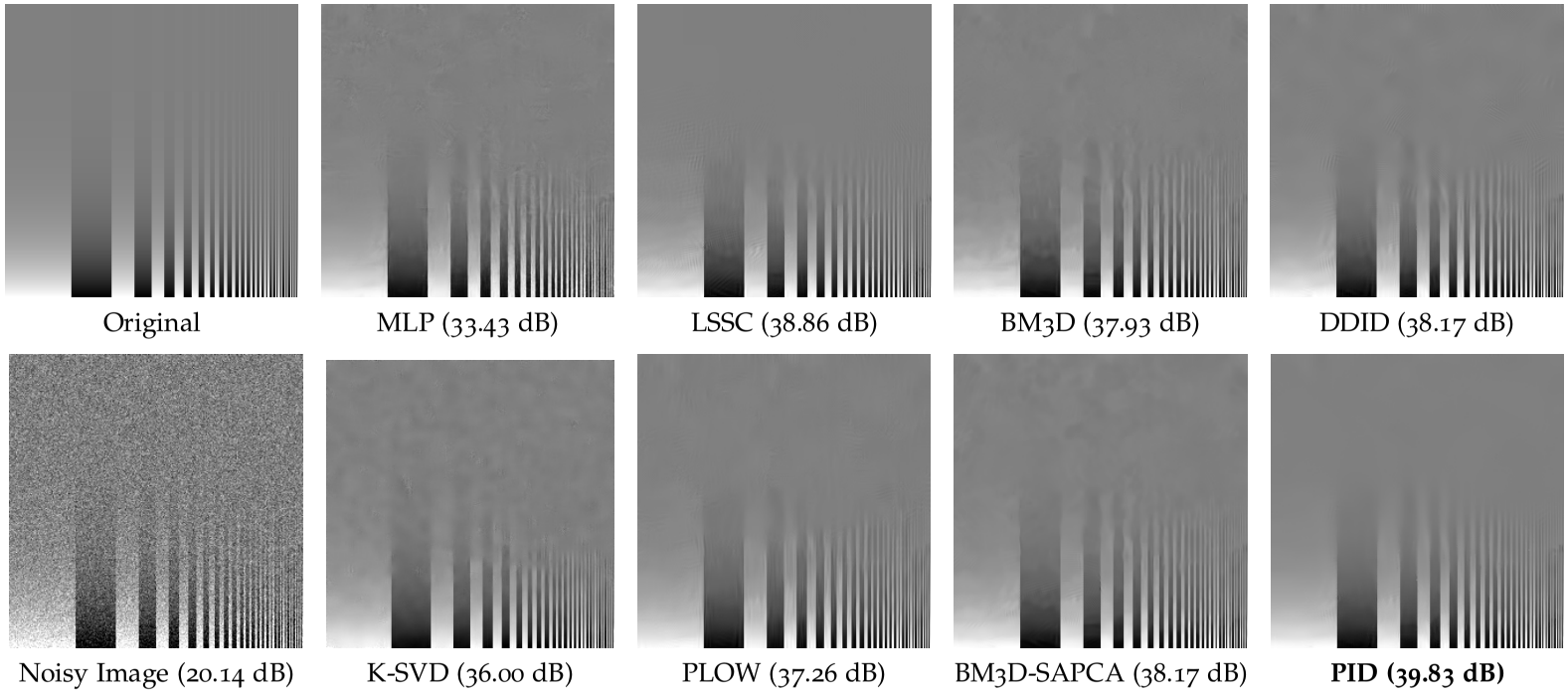


Figure 6.12: *Comparison of denoising* CSF *charts with rectangular waves. Noise sigma is* $\sigma = 25$*. This table exposes the characteristic artifacts of each method. Most methods suffer either from loss of contrast or low-frequency noise in homogeneous regions. The results of* PID *are nearly artifact-free.*

6.6 Conclusions

In this chapter, we have developed PID, an improvement over the previous denoiser DDID. Unlike other state-of-the-art methods, the resulting images of PID are nearly artifact free and produce visually pleasing results for both natural and synthetic images.

In contrast to DDID, PID is mostly unguided and resembles a physical process. Unlike DDID, which uses an oracle and the noisy image to calculate an improved oracle, PID continuously improves the oracle. Using deterministic annealing, we change the shape of the robust noise estimator over time. The scale parameter of the range kernel corresponds to the falling temperature in deterministic and simulated annealing. This annealing process allows us to find near-optimal solutions.

Compared to DDID, PID uses more iterations. As they are both based on DDF, they both have the FFT as the bottleneck. Our focus in this work is to emphasize on the quality and simplicity of the denoising process. Some performance improvements are quite straightforward. For example, it would be possible to adaptively grow the window radius to follow the growing spatial kernel. We use this in the implementation of our next and final denoising method, DDID2.

7

Dual-Domain Image Denoising Revised

Fail, fail again.
Fail better.

Samuel Beckett

In the previous two chapters, we have developed DDID and PID, two denoising methods which are complementary. DDID iterates using the previously filtered image as a guide, while PID iterates without guidance. We note that as a last step of PID we still had to use a guided DDID step to get optimal results. From these observations, we infer that a combination of both methods should be better than DDID and simpler to implement than PID.

We formulate our new iterative denoiser similar to DDID, by stating the guided iteration

$$x_{n-1} = y - \operatorname{Im} \mathrm{DDF}_n(x_n + iy), \tag{7.1}$$

which is a time reversed version of Equation 5.1. As opposed to DDID, where we empirically found our scale parameters, we use the experience from PID and define the scale parameters as functions of iteration number n, counting down from N to 1. We define the scale parameters for the bilateral kernel function $k(\cdot)$ and the frequency kernel function $K(\cdot)$ as

$$S_n = 2\,\sigma_s^2\,\alpha^{\frac{1-n}{2N}} \tag{7.2}$$

$$T_n = \gamma_r\,\sigma^2\,\alpha^{\frac{n-1}{N}} \tag{7.3}$$

$$V = \gamma_f\,\sigma^2. \tag{7.4}$$

As n counts down to 1, the spatial scale S_n becomes larger, while the spatial range factor T_n becomes smaller. The base α controls the initial range scale T_N

and the final spatial scale S_1. The frequency range scale V remains constant. We achieved the best results using the constants

$$N = 8 \tag{7.5}$$
$$\sigma_s = 13 \tag{7.6}$$
$$\gamma_r = 5.3/N \tag{7.7}$$
$$\gamma_f = 13/N \tag{7.8}$$
$$\alpha = e^{15}. \tag{7.9}$$

For PID, we had used a fixed kernel $r = 15$, and an even larger kernel for the final DDID step using $r = 31$. If we fixed our kernel radius to $r = 31$, every iteration would be unnecessarily expensive. To improve performance, we adapt the kernel radius r to be twice the spatial standard deviation $\sqrt{S_n/2}$ and at least 4 pixels large. The kernel radius r is thus

$$r = \max\left(4, \operatorname{round}(2\sqrt{S_n/2})\right). \tag{7.10}$$

In addition to dynamic scale parameters, we change also the shape of the range kernels. We found that Gaussians work better in the beginning, and kernels with strong outlier rejection like the Epanechnikov kernel, or the similar cosine kernel work better against the end of the iterations. Since the Gaussian can be approximated by powers of cosines, we use this relationship to define our dynamic range kernel for both domains. The resulting spatial and frequency kernel functions are

$$k(d^2, \rho^2) = \cos\left(\min\left(\frac{\pi}{2}, \sqrt{\frac{d^2}{T_n\, n}}\right)\right)^n e^{-\frac{\rho^2}{S_n}} \tag{7.11}$$
$$K(D^2) = \cos\left(\min\left(\frac{\pi}{2}, \sqrt{\frac{D^2}{V\, n}}\right)\right)^n. \tag{7.12}$$

As visualized in Figure 7.1, the range kernel starts as an approximation of a Gaussian, becomes steeper over time, and ends as a cosine. The spatial range kernel also becomes narrower over time. Figure 7.2 depicts the evolution of the bilateral kernel from the side.

We also specify the confidence factors dynamically. In the beginning, due to the large scale parameters, the spatially estimated noise is biased and cannot be trusted. Over time, the noise estimate becomes more accurate until in the end, when it can be fully trusted. The same applies for the estimated noise in

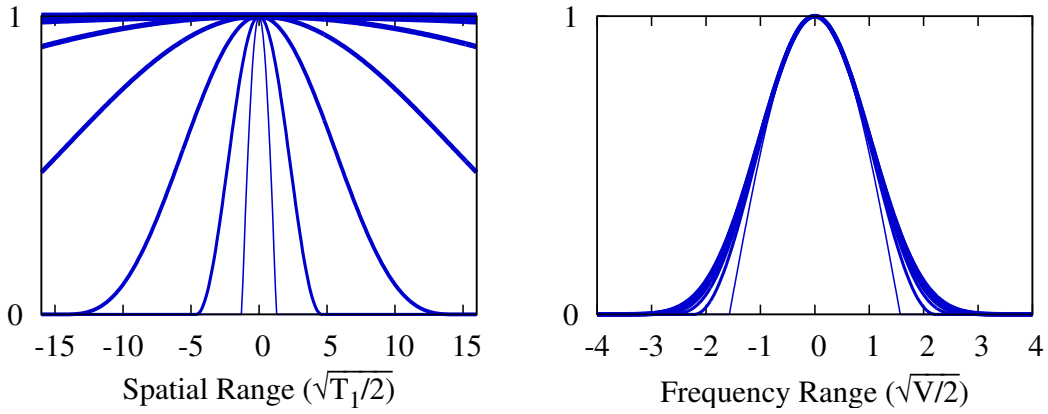


Figure 7.1: *Evolution of the range kernels from thick to thin lines as n counts down from N to 1 in Equation 7.11 and Equation 7.12. The range kernel functions become steeper over the iterations. The spatial range kernel function additionally becomes narrower. For reference, the spatial range kernel function is normalized by the final standard deviation $\sqrt{T_1/2}$ and the frequency range kernel function is normalized by the standard deviation $\sqrt{V/2}$.*

the frequency domain. We therefore specify the confidence factors a and A to follow a sine ramp from 0 to 1, expressed as

$$a = A = \cos\frac{n-1}{N}\frac{\pi}{2}. \tag{7.13}$$

The differences to the previous methods DDID and PID are the following. DDID uses a fixed scale σ_s for the spatial Gaussian and the role of the spatial and frequency range parameters γ_r and γ_f over the iterations were empirically chosen. PID uses scale parameters similar to equation Equation 7.2 and Equation 7.3 due to deterministic annealing. However, PID does not change the shape of the range kernel as we propose in Equation 7.11 and Equation 7.12. More importantly, the PID formulation is not guided and requires 30 iterations and an additional guided DDID step. Our new formulation only requires 8 guided iterations and is almost as simple in implementation as DDID.

Figure 7.3, Figure 7.4, and Figure 7.5, Figure 7.6 visually compare our new denoiser, DDID2, against other state-of-the-art denoising methods that support color images. DDID2 and PID produce the cleanest and smoothest results ans work equally well on natural and synthetic images. Table 10.3, Table 10.4, and Table 10.5 summarize the numerical results. Our method matches PID in quality, but requires only a third of the iterations without a separate guided DDID step. For grayscale images, DDID2 starts competing with the best denoisers. For color images, DDID2 and PID produce the best results.

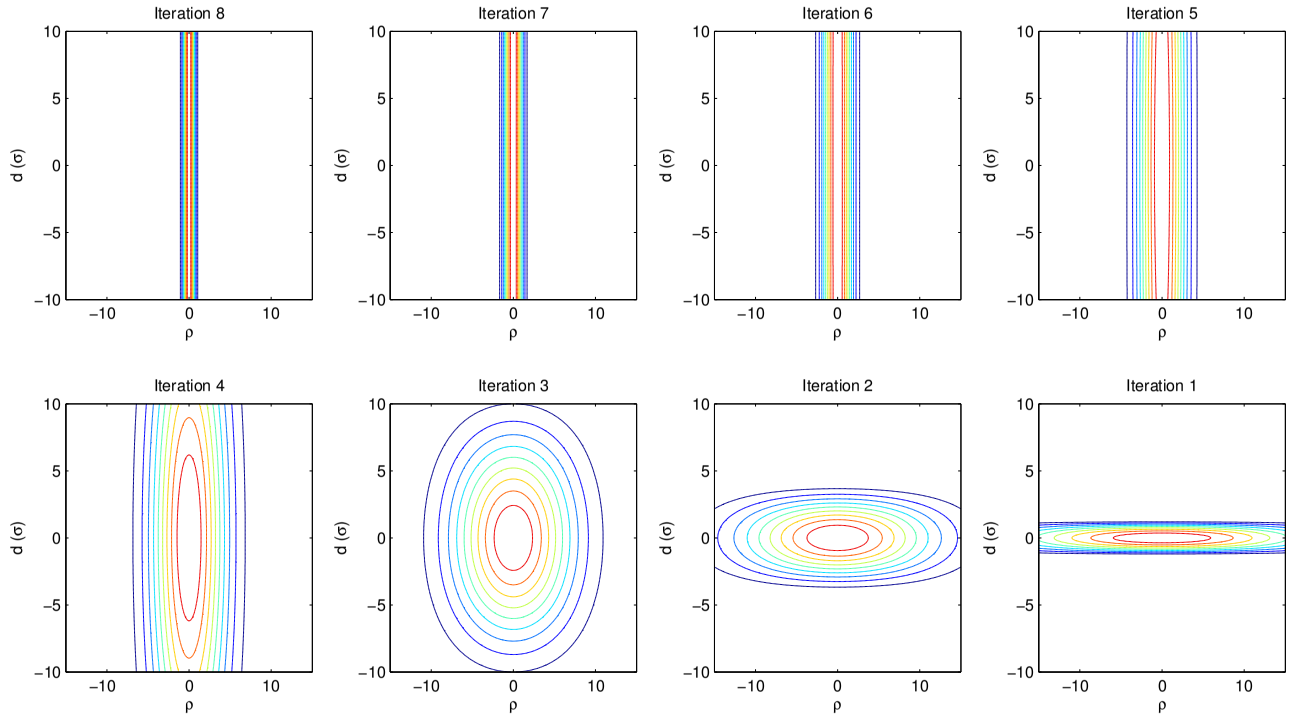
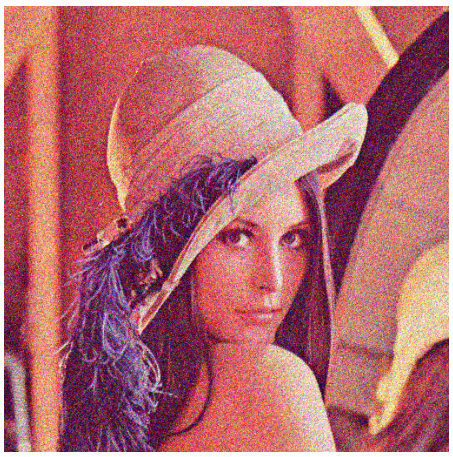


Figure 7.2: *Evolution of the bilateral kernel over 8 iterations. The shown range gradient d is normalized by the noise sigma σ. The unit of the spatial distance ρ is pixels. The bilateral kernel starts tall and thin and ends flat and wide. The final range scale of the kernel is close to the noise sigma σ and the spatial scale spans the entire window.*

Original



Noisy Image (16.09 dB)

DDID (30.68 dB)

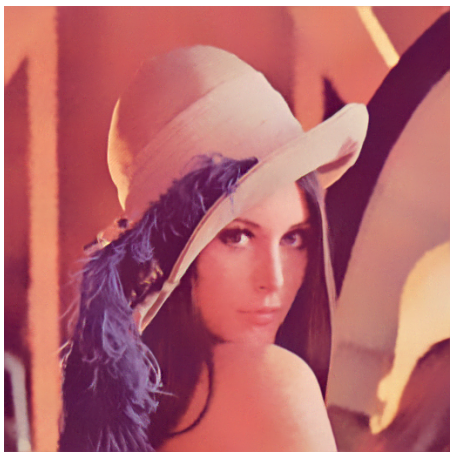
BM3D (30.11 dB)

DDID2 (30.87 dB)

PID (30.87 dB)

Figure 7.3: *Comparison of denoising a natural color image with* $\sigma = 40$.

NLB (30.42 dB)



NLM (28.87 dB)

K-SVD (29.66 dB)

PLOW (29.26 dB)

TV (27.21 dB)

FOE (28.47 dB)

Figure 7.4: *Comparison of denoising a natural color image with* $\sigma = 40$.

Original

Noisy Image (16.10 dB)

DDID (30.26 dB)

BM3D (28.06 dB)

DDID2 (30.63 dB)

PID (30.84 dB)

Figure 7.5: *Comparison of denoising a synthetic color image with noise sigma $\sigma = 40$. Our methods preserve edges well and provide cleaner results.* DDID *shows some artifacts, which are mostly removed by* DDID2. PID *has the fewest artifacts.* BM3D *suffers from residual noise.*

NLB (29.31 dB)

NLM (28.29 dB)

K-SVD (27.97 dB)

PLOW (26.40 dB)

TV (26.24 dB)

FOE (25.34 dB)

Figure 7.6: *Other methods suffer from residual noise or blur. For this image,* NLM *produces a better result than* BM3D.

7.1 Conclusions

Our last image denoiser, DDID2, combines the best parts of DDID and PID. It produces results matching the quality of PID, while being structurally as simple as DDID. In the process of defining this denoiser, we found two new insights. First, the spatial and frequency kernel functions should not only change in scale but also in shape. Second, the noise estimates in the spatial and frequency domains are initially not reliable and become more reliable over time.

In the next chapter, we will extend the application domain of DDF outside of classic image denoising and show its power for removing denoising and compression artifacts.

8 Artifact Removal

Do something right or something new.

Anonymous

In this chapter, we show two different applications using the DDF. The first application is the removal of residual noise from other common image denoising algorithms. While it is not surprising that older algorithms produce artifacts, even state-of-the-art methods like BM3D suffer from residual noise. We show in Section 8.1 how DDF can be used to remove denoising artifacts. The second application is deblocking of JPEG compressed images. For high compression ratio, JPEG produces grid-aligned discontinuities over the whole image and ringing artifacts near edges. Such artifacts have been addressed by many tailored methods. We show in Section 8.2 that DDF is also capable of deblocking JPEG images.

8.1 Noise Artifact Removal

To remove the artifacts of a denoising method, we post-process the denoised output with DDF by feeding it as the guide image g to filter the noisy image y, i.e., $x = y - \operatorname{Im}\mathrm{DDF}(g + iy)$. We configure DDF using the following confidence factors and kernel functions:

$$a = A = 1 \tag{8.1}$$

$$k(d^2, \rho^2) = e^{-\frac{d^2}{\gamma_r \sigma^2}} e^{-\frac{\rho^2}{2\sigma_s^2}} \tag{8.2}$$

$$K(D^2) = \max\left(0, 1 - \frac{D^2}{\gamma_f \sigma^2}\right). \tag{8.3}$$

Here, the spatial kernel function $k(\cdot,\cdot)$ is the ordinary bilateral kernel with the spatial scale parameter σ_s, noise variance σ^2, and a spatial range parameter γ_r. For the frequency kernel function $K(\cdot)$, we chose the Epanechnikov estimator, introducing the frequency range parameter γ_f.

For grayscale images, we post-process the output of popular denoising methods for noise sigma $\sigma \in \{10, 25, 40\}$ and set the parameters to $r = 15$, $\sigma_s = 7$, $\gamma_r = 0.7$, and $\gamma_f = 2.3$. For color images, the input images had noise sigma $\sigma \in \{20, 40\}$ and we changed the range parameters to $\gamma_r = 0.4$ and $\gamma_f = 1.1$.

For grayscale images, we post-processed the output of the denoising methods K-LLD [50], K-SVD [38], PLOW [28], non-local means (NLM) [20], non-local Bayes (NLB) [27], LSSC [34], multi-layer perceptrons (MLP) [35], BM3D [26], BM3D-SAPCA [31], and SAIST [51]. For color images, we used the output of the methods supporting colors, BM3D, NLB, NLM, and PLOW.

Figure 8.1 and Figure 8.2 give visual examples for removing denoising artifacts. Low-frequency noise of K-SVD, graininess of NLM, outliers of NLB, and wavy patterns of LSSC: all these artifacts are reduced or removed by DDF. Table 8.1 and Table 8.2 show that nearly every grayscale output of any method can be numerically improved by post-processing with DDF. The more smooth regions an image has, the larger the gain is in PSNR. This is not surprising, since most methods excel at denoising natural images, whereas they have difficulties denoising synthetic images where smooth regions dominate. Images denoised by SAIST show almost no artifacts and can only be improved for high-noise situations where the signal is more homogeneous. As can be seen in Table 8.3, for noisy color images with $\sigma = 40$, all images show improvement.

8.2 JPEG Artifact Removal

For JPEG deblocking, we only have the artifact contaminated image, so we identify the guide image g with the noisy image y. We compressed grayscale and color images using three quality settings in MATLAB, $Q \in \{30, 20, 10\}$ and used a corresponding noise sigma $\sigma \in \{20, 25, 40\}$. For grayscale images, we used the parameters $r = 15$, $\sigma_s = 7$, $\gamma_r = 1.7$, $\gamma_f = 1.1$, and chose the noise sigma $\sigma = 25$. For color images, we changed $\gamma_r = 2.8$ and $\gamma_f = 4.2$. We compare our results against SA-DCT, a state-of-the-art JPEG deblocker by Foi et al. [41]. We also compare against the bilaterally filtered image, using the same parameters as DDF.

Figure 8.3 shows the deblocking of a JPEG image. The removed artifacts are

the typical block patterns. The image improves almost everywhere. Table 8.4 numerically summarizes the results for deblocking JPEG images. For grayscale images, DDF approaches the quality of SA-DCT. For color images, the results are nearly identical.

8.3 Conclusions

We have shown that DDF can be used as a stand-alone filter to remove various types of artifacts. We suspect that DDF is a universal tool that can be deployed in many applications where edges and details must be preserved.

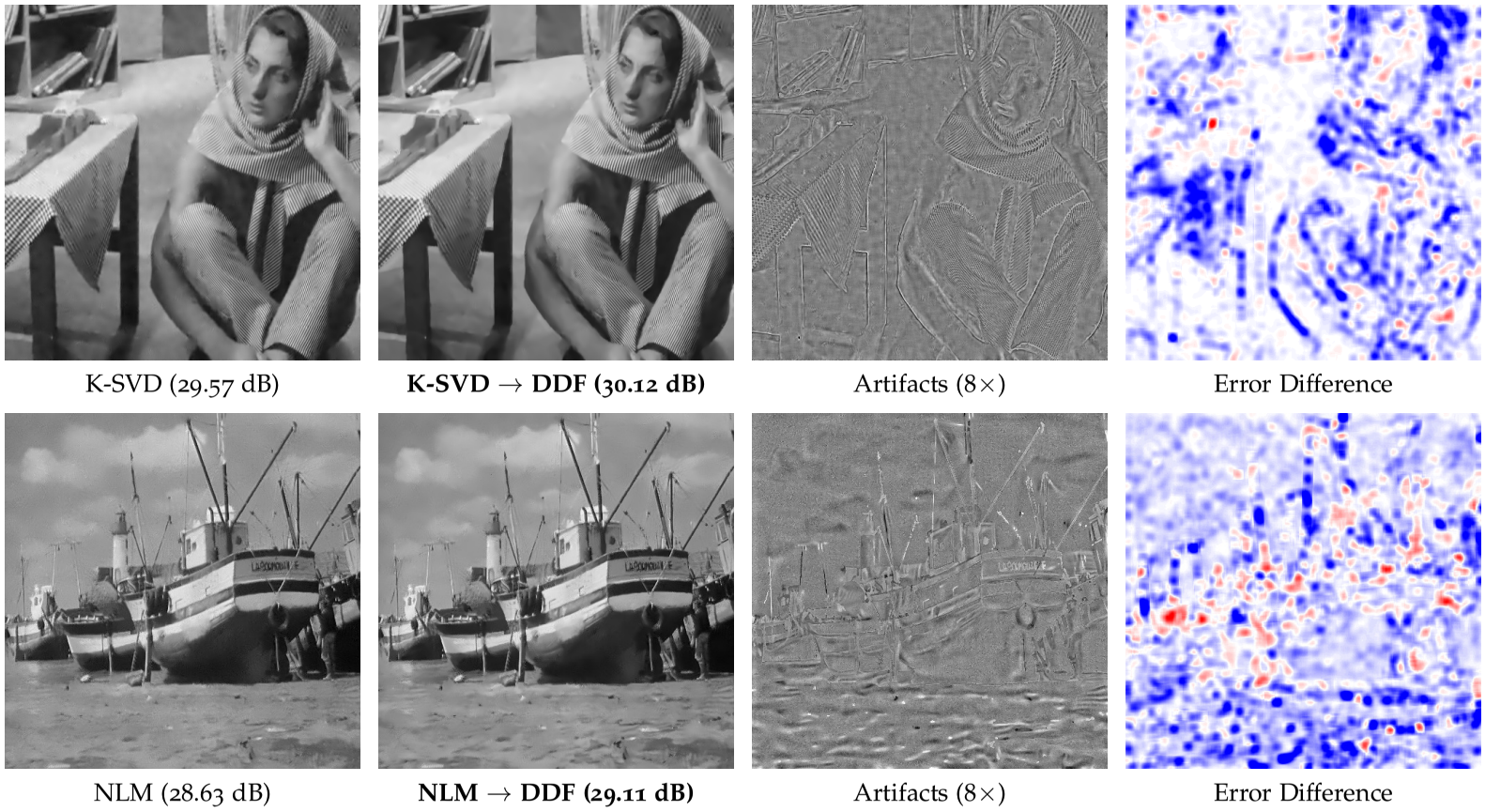
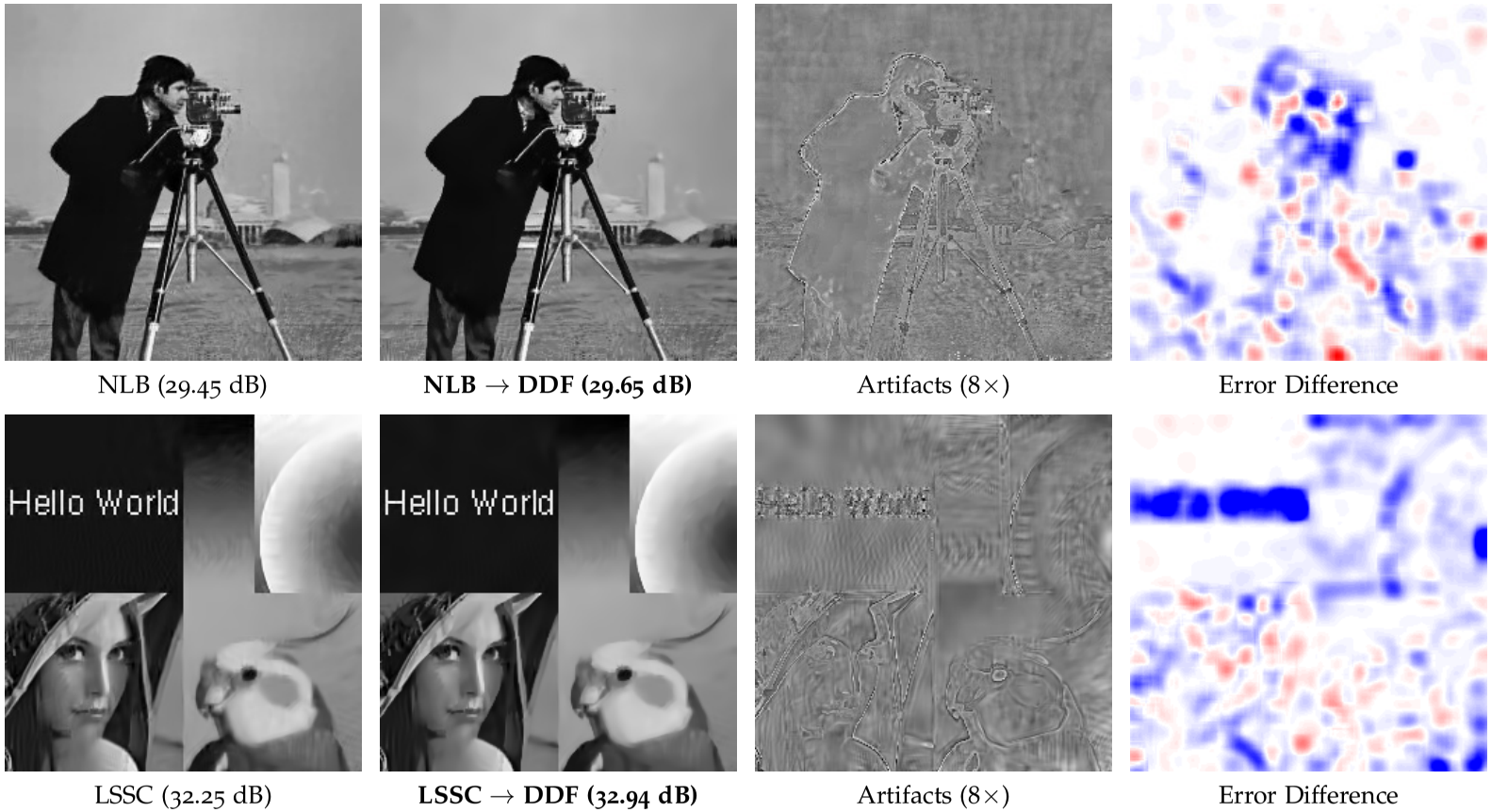


Figure 8.1: DDF *removes artifacts like low-frequency noise and graininess. The noise sigma is* $\sigma = 25$. *The artifacts images are the difference images between before and after processing with* DDF. *In the error difference images, red and blue mark where the error increased and decreased respectively.*



NLB (29.45 dB) | **NLB → DDF (29.65 dB)** | Artifacts (8×) | Error Difference

LSSC (32.25 dB) | **LSSC → DDF (32.94 dB)** | Artifacts (8×) | Error Difference

Figure 8.2: DDF *removes artifacts removes artifacts like outliers and ringing. The noise sigma is* $\sigma = 25$. *The artifacts images are the difference images between before and after processing with* DDF. *In the error difference images, red and blue mark where the error increased and decreased respectively.*

Grayscale	SAIST	SAPCA	BM3D	MLP	LSSC
Barbara	**35.19** → 34.93	**35.10** → 34.90	**34.98** → 34.83	**34.07** → 34.01	**34.99** → 34.98
Boats	**33.93** → 33.79	**34.10** → 34.00	**33.92** → 33.87	**33.81** → 33.80	**34.03** → 34.01
Cameraman	**34.23** → 34.09	**34.59** → 34.48	34.18 → **34.27**	34.18 → **34.22**	34.24 → **34.32**
Couple	**34.00** → 33.91	**34.17** → 34.14	34.04 → **34.05**	33.91 → **33.98**	34.01 → **34.05**
Finger Print	**32.68** → 32.48	**32.64** → 32.42	**32.46** → 32.32	**32.57** → 32.45	32.57 → **32.60**
Hill	**33.70** → 33.60	**33.83** → 33.80	33.62 → **33.67**	33.59 → **33.63**	33.67 → **33.70**
House	**36.81** → 36.71	**37.01** → 36.92	36.71 → **36.76**	35.98 → **36.12**	36.95 → **36.96**
Lena	**35.85** → 35.69	**36.07** → 35.96	**35.93** → 35.87	35.85 → **35.86**	35.85 → **35.92**
Man	**34.13** → 34.05	**34.25** → 34.23	33.98 → **34.07**	34.11 → **34.17**	34.10 → **34.15**
Montage	37.19 → **37.24**	**37.85** → 37.81	37.35 → **37.53**	36.51 → **37.12**	37.26 → **37.50**
Pepper	**34.79** → 34.72	**34.94** → 34.89	34.68 → **34.74**	34.72 → **34.82**	34.80 → **34.85**

$\sigma = 10$

Grayscale	SAIST	SAPCA	BM3D	MLP	LSSC
Barbara	**31.23** → 31.07	**31.00** → 30.91	30.72 → **30.74**	29.55 → **29.74**	30.49 → **30.70**
Boats	**29.97** → 29.95	**30.03** → 30.01	29.91 → **29.95**	29.97 → **30.00**	29.90 → **30.00**
Cameraman	29.41 → **29.55**	29.81 → **29.88**	29.45 → **29.68**	29.61 → **29.85**	29.50 → **29.84**
Couple	29.74 → **29.81**	29.82 → **29.89**	29.72 → **29.83**	29.74 → **29.84**	29.67 → **29.84**
Finger Print	**27.93** → 27.73	**27.81** → 27.62	**27.70** → 27.61	**27.65** → 27.49	**27.62** → 27.60
Hill	**29.90** → 29.88	29.96 → 29.96	29.85 → **29.89**	29.88 → 29.88	29.83 → **29.92**
House	**33.17** → 33.10	32.96 → **33.00**	32.86 → **33.02**	32.57 → **32.80**	33.13 → **33.15**
Lena	**32.26** → 32.25	32.23 → 32.23	32.08 → **32.19**	**32.26** → 32.22	31.86 → **32.16**
Man	29.75 → **29.80**	29.81 → **29.87**	29.62 → **29.76**	29.89 → **29.95**	29.70 → **29.83**
Montage	32.35 → **32.80**	32.97 → **33.25**	32.37 → **32.99**	32.04 → **32.68**	32.25 → **32.94**
Pepper	30.40 → **30.58**	30.43 → **30.52**	30.16 → **30.44**	30.31 → **30.64**	30.23 → **30.49**

$\sigma = 25$

Grayscale	SAIST	SAPCA	BM3D	MLP	LSSC
Barbara	28.62 → 28.62	**28.68** → 28.66	27.99 → **28.20**	27.62 → **27.75**	28.17 → **28.44**
Boats	27.62 → **27.69**	27.92 → **27.97**	27.74 → **27.84**	**28.54** → 28.33	27.77 → **27.91**
Cameraman	27.29 → **27.64**	27.57 → **27.70**	27.18 → **27.50**	28.08 → **28.16**	27.34 → **27.79**
Couple	27.33 → **27.46**	27.58 → **27.65**	27.48 → **27.60**	**28.24** → 28.05	27.41 → **27.58**
Finger Print	**25.55** → 25.33	**25.54** → 25.30	**25.30** → 25.22	**25.89** → 25.50	25.30 → 25.30
Hill	27.87 → **27.90**	28.08 → **28.11**	27.99 → **28.04**	**28.65** → 28.38	28.00 → **28.07**
House	**31.38** → 31.28	30.75 → **30.99**	30.65 → **30.90**	**31.19** → 31.16	31.10 → **31.19**
Lena	30.03 → **30.21**	30.10 → **30.22**	29.86 → **30.12**	**30.83** → 30.53	29.91 → **30.16**
Man	27.58 → **27.70**	27.83 → **27.90**	27.65 → **27.79**	**28.48** → 28.28	27.64 → **27.83**
Montage	29.50 → **30.33**	30.02 → **30.68**	29.52 → **30.42**	30.40 → **30.84**	29.43 → **30.50**
Pepper	27.94 → **28.23**	28.10 → **28.23**	27.70 → **28.02**	28.54 → **28.72**	27.86 → **28.25**

$\sigma = 40$

Table 8.1: PSNR *(dB) improvement of grayscale images with* DDF *(first out of two sets of methods). With the exception of high-quality methods like* SAIST *or* BM3D-SAPCA*, and low-noise scenarios,* DDF *consistently removes artifacts and improves* PSNR *values.*

Grayscale	NLB	NLM	PLOW	K-SVD	K-LLD
Barbara	**34.80** → 34.67	**33.14** → 33.12	33.79 → **34.16**	34.43 → **34.56**	33.11 → **33.34**
Boats	**33.87** → 33.80	**32.89** → 32.88	32.97 → **33.31**	33.63 → **33.78**	33.00 → **33.33**
Cameraman	**34.39** → 34.31	33.40 → **33.52**	33.17 → **33.61**	33.75 → **33.98**	32.81 → **33.18**
Couple	**33.97** → **33.97**	32.88 → **32.89**	33.12 → **33.51**	33.54 → **33.79**	33.10 → **33.47**
Finger Print	**32.41** → 32.21	**30.98** → 30.77	31.03 → **31.63**	32.39 → **32.50**	31.65 → **31.70**
Hill	**33.70** → 33.68	**32.81** → 32.79	32.62 → **32.98**	33.36 → **33.55**	32.78 → **33.02**
House	**36.26** → 36.25	34.90 → **35.13**	36.22 → **36.58**	35.95 → **36.27**	35.24 → **35.59**
Lena	35.73 → **35.74**	34.29 → **34.45**	35.30 → **35.57**	35.48 → **35.70**	35.26 → **35.43**
Man	**34.11** → 34.09	33.07 → **33.09**	32.95 → **33.42**	33.59 → **33.87**	33.18 → **33.56**
Montage	37.20 → **37.39**	35.23 → **35.61**	36.12 → **36.99**	36.08 → **36.78**	35.43 → **36.44**
Pepper	**34.80** → 34.77	33.44 → **33.53**	33.56 → **34.16**	34.24 → **34.51**	33.85 → **34.20**

$\sigma = 10$

Grayscale	NLB	NLM	PLOW	K-SVD	K-LLD
Barbara	30.25 → **30.30**	28.95 → **29.44**	30.20 → **30.45**	29.56 → **30.12**	27.69 → **28.26**
Boats	29.67 → **29.77**	28.63 → **29.11**	29.53 → **29.76**	29.31 → **29.64**	29.26 → **29.65**
Cameraman	29.45 → **29.65**	28.70 → **29.13**	28.65 → **29.23**	28.90 → **29.50**	28.42 → **29.08**
Couple	29.38 → **29.57**	28.27 → **28.88**	29.35 → **29.64**	28.89 → **29.38**	29.14 → **29.49**
Finger Print	**27.53** → 27.33	**26.12** → **26.12**	27.05 → **27.13**	27.25 → **27.50**	**26.97** → **26.97**
Hill	29.62 → **29.76**	28.67 → **29.21**	29.60 → **29.75**	29.22 → **29.52**	29.25 → **29.54**
House	32.40 → **32.57**	31.18 → **31.91**	32.72 → **33.03**	32.07 → **32.76**	31.49 → **32.35**
Lena	31.79 → **31.93**	30.41 → **31.13**	31.90 → **32.13**	31.35 → **31.84**	31.42 → **31.90**
Man	29.62 → **29.76**	28.60 → **29.13**	29.33 → **29.62**	29.11 → **29.49**	29.27 → **29.63**
Montage	31.97 → **32.53**	30.70 → **31.71**	30.95 → **32.71**	31.16 → **32.30**	30.52 → **31.82**
Pepper	30.12 → **30.31**	28.69 → **29.46**	29.63 → **30.23**	29.70 → **30.30**	29.56 → **30.20**

$\sigma = 25$

Grayscale	NLB	NLM	PLOW	K-SVD	K-LLD
Barbara	28.07 → **28.26**	26.65 → **27.76**	28.10 → **28.37**	26.89 → **27.60**	24.84 → **25.87**
Boats	27.39 → **27.61**	26.27 → **27.15**	27.63 → **27.86**	27.06 → **27.44**	26.33 → **27.23**
Cameraman	27.17 → **27.53**	26.49 → **27.12**	26.66 → **27.32**	26.76 → **27.46**	25.58 → **26.63**
Couple	27.18 → **27.46**	25.66 → **26.80**	27.32 → **27.58**	26.39 → **26.91**	26.19 → **27.01**
Finger Print	**25.39** → 25.27	24.07 → **24.64**	**25.20** → 25.12	24.69 → **25.04**	24.00 → **24.28**
Hill	27.75 → **27.95**	26.45 → **27.53**	27.89 → **28.04**	27.21 → **27.58**	26.53 → **27.43**
House	30.26 → **30.67**	28.83 → **30.18**	30.55 → **31.05**	29.58 → **30.50**	27.44 → **29.04**
Lena	29.81 → **30.09**	28.23 → **29.53**	29.86 → **30.20**	29.05 → **29.73**	27.63 → **29.02**
Man	27.51 → **27.74**	26.35 → **27.34**	27.52 → **27.80**	27.04 → **27.47**	26.42 → **27.34**
Montage	29.11 → **30.07**	27.67 → **29.48**	27.90 → **30.06**	28.64 → **29.93**	26.71 → **28.40**
Pepper	27.69 → **28.10**	25.73 → **27.27**	27.50 → **28.09**	27.40 → **28.03**	26.15 → **27.23**

$\sigma = 40$

Table 8.2: PSNR *(dB) improvement of grayscale images with* DDF *(second out of two sets of methods). For this set of methods, except for* NLB *at low noise sigma,* DDF *consistently removes artifacts and improves* PSNR *values.*

Color	BM3D	NLB	NLM	PLOW
Baboon	25.95 → **26.10**	**26.53** → 26.44	25.65 → **25.98**	24.22 → **24.96**
F-16	32.77 → **33.00**	**33.03** → 33.00	31.31 → **32.21**	30.97 → **32.23**
House	**33.02** → 32.81	**32.65** → 32.59	31.39 → **31.99**	31.91 → **32.53**
Kodak 1	29.12 → **29.14**	29.29 → **29.32**	27.49 → **28.43**	25.68 → **26.60**
Kodak 2	**32.44** → 32.26	32.28 → **32.30**	30.69 → **31.49**	29.86 → **31.21**
Kodak 3	**34.56** → 34.51	34.49 → **34.50**	32.33 → **33.39**	30.46 → **32.51**
Kodak 12	**33.76** → 33.52	**33.37** → 33.28	31.62 → **32.36**	30.02 → **31.92**
Lena	32.27 → 32.27	**32.25** → 32.22	30.88 → **31.53**	31.00 → **31.73**
Pepper	31.22 → **31.27**	31.23 → 31.23	30.28 → **30.83**	30.37 → **31.00**
Lake	28.68 → **28.87**	**29.10** → 29.08	28.29 → **28.58**	27.60 → **28.27**
Tiffany	32.31 → **32.41**	32.37 → **32.53**	31.13 → **31.91**	31.51 → **32.08**

$\sigma = 25$

Color	BM3D	NLB	NLM	PLOW
Baboon	23.87 → **24.07**	**24.50** → 24.47	23.22 → **23.91**	22.56 → **23.17**
F-16	30.24 → **30.84**	30.87 → **30.94**	28.61 → **30.07**	29.05 → **30.37**
House	30.59 → **31.00**	30.85 → **30.99**	29.05 → **30.43**	30.04 → **30.91**
Kodak 1	26.59 → **26.68**	26.84 → **26.97**	24.76 → **25.67**	24.32 → **25.29**
Kodak 2	30.30 → **30.41**	30.24 → **30.37**	28.36 → **29.54**	28.22 → **29.75**
Kodak 3	31.59 → **31.88**	32.08 → **32.13**	29.96 → **31.03**	28.46 → **30.69**
Kodak 12	31.32 → **31.38**	**31.26** → 31.20	29.41 → **30.49**	28.04 → **30.29**
Lena	30.11 → **30.47**	30.48 → **30.54**	28.90 → **29.85**	29.26 → **30.15**
Pepper	29.32 → **29.77**	29.65 → **29.76**	28.19 → **29.34**	28.72 → **29.64**
Lake	26.88 → **27.22**	27.43 → **27.44**	26.04 → **26.63**	26.03 → **26.74**
Tiffany	30.24 → **30.50**	30.40 → **30.68**	28.71 → **30.00**	29.84 → **30.44**

$\sigma = 40$

Table 8.3: PSNR *(dB) improvement of color images before and after post-processing with* DDF. *With the exception of* BM3D *for low-noise scenarios,* DDF *consistently removes artifacts and improves* PSNR *values.*

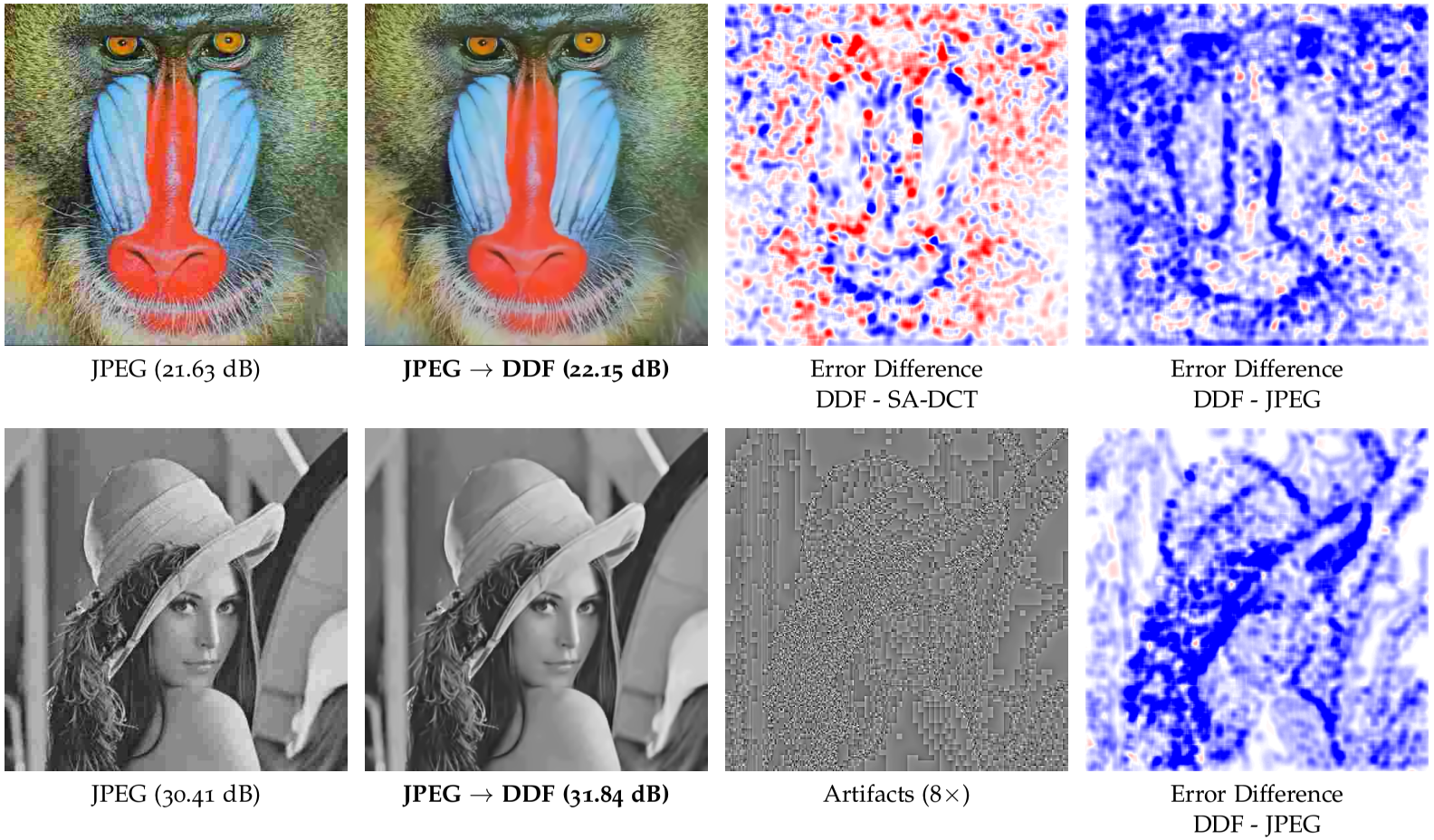


Figure 8.3: DDF *removes* JPEG *block artifacts. The* JPEG *compression is set to* MATLAB *quality* $Q = 10$*. The artifacts image is the difference image between before and after processing with* DDF*. In the error difference images, red and blue mark where the error increased and decreased respectively.*

Grayscale	JPEG	BF	DDF	SA-DCT	JPEG	BF	DDF	SA-DCT	JPEG	BF	DDF	SA-DCT
Barbara	30.16	29.65	**31.09**	30.72	28.25	27.83	**29.26**	28.90	25.70	25.15	**26.95**	26.54
Boats	31.83	30.25	32.48	**32.54**	30.49	28.88	31.23	**31.28**	28.13	26.34	29.09	**29.14**
Cameraman	29.93	29.96	**30.70**	30.69	28.59	28.62	29.37	**29.39**	26.47	26.20	27.33	**27.48**
Couple	31.75	30.02	32.39	**32.44**	30.41	28.55	31.18	**31.24**	28.05	25.81	28.98	**29.04**
Finger Print	31.16	29.61	31.71	**32.05**	29.49	27.61	30.20	**30.54**	26.57	23.61	27.52	**27.80**
Hill	32.04	30.27	**32.55**	32.53	30.82	28.95	**31.45**	31.42	28.61	26.55	**29.43**	29.40
House	34.20	33.18	**35.19**	35.07	33.02	31.92	34.09	**34.10**	30.56	28.82	31.93	**32.09**
Lena	34.28	32.42	35.09	**35.12**	32.96	31.19	34.01	**34.04**	30.41	28.66	**31.84**	**31.84**
Man	31.80	30.33	32.50	**32.55**	30.55	29.00	31.34	**31.39**	28.27	26.62	29.25	**29.30**
Montage	32.76	33.18	34.13	**34.15**	31.24	31.53	32.61	**32.63**	28.56	28.51	30.04	**30.14**
Pepper	31.63	30.91	32.64	**32.74**	30.29	29.50	31.38	**31.52**	27.82	26.69	29.10	**29.30**
			$Q = 30$				$Q = 20$				$Q = 10$	

Color	JPEG	BF	DDF	SA-DCT	JPEG	BF	DDF	SA-DCT	JPEG	BF	DDF	SA-DCT
Baboon	23.85	24.08	**24.18**	24.07	23.07	23.27	**23.44**	23.38	21.63	21.74	**22.15**	22.13
F-16	30.06	30.99	**31.20**	31.08	28.90	29.92	**30.17**	30.12	26.87	28.01	**28.33**	28.30
House	28.96	**29.91**	29.85	29.78	27.87	**28.83**	28.81	28.77	26.25	27.22	27.48	**27.53**
Kodak 1	28.21	28.34	28.77	**28.83**	26.94	27.05	27.57	**27.63**	24.77	24.56	25.48	**25.52**
Kodak 2	31.37	31.15	**31.99**	31.83	30.01	29.91	**30.70**	30.64	27.85	27.84	**28.67**	28.63
Kodak 3	32.86	33.21	**34.00**	**34.00**	31.44	31.96	**32.68**	**32.68**	28.56	29.10	29.81	**29.85**
Kodak 12	32.81	32.60	**33.62**	33.61	31.33	31.39	**32.30**	32.26	28.71	29.18	**29.81**	29.76
Lake	26.84	27.43	**27.57**	27.38	26.07	26.76	**26.92**	26.79	24.39	25.02	**25.37**	25.34
Lena	30.91	31.20	**31.87**	31.79	29.83	30.29	**31.01**	31.00	27.53	28.20	29.00	**29.06**
Pepper	28.40	29.01	**29.19**	29.14	27.57	28.32	**28.54**	**28.54**	25.77	26.72	27.03	**27.12**
Tiffany	29.21	29.50	**29.80**	29.64	28.40	28.74	**29.11**	29.00	26.83	27.37	**27.79**	27.78
			$Q = 30$				$Q = 20$				$Q = 10$	

Table 8.4: PSNR *(dB) comparison of* JPEG *deblocking with* DDF. *For grayscale images,* DDF *approaches* SA-DCT *in quality. For color images, their results are similar.*

9 Implementation

Life as we know it would be very different without the FFT.

Charles van Loan

In this chapter, we give executable implementations of the algorithms described in the previous chapters using MATLAB. Expressing our algorithms using MATLAB results in concise code as presented in this chapter. Not all operations are efficient, so we give hints for optimization.

9.1 Border Handling

Our filter uses pixels from its neighborhood. However, near the border of the image, the neighborhood pixels are not available. There are two choices, either adapt the filter to not access pixels outside the defined image range, or to enlarge the image with synthesized pixels. An example for the first case is SA-DCT, which by design adapts the shape to its environment by reducing the filter size near the border of the image. For DDF, we can achieve a similar effect using the bilateral filter, by padding with extreme values, which produce bilateral weights close to 0 for pixels outside the image.

An easier way than adapting the filter to handle borders is to use padding strategies. A common strategy is to use symmetric padding, which works well in most cases. For some images, symmetric padding may introduce wave artifacts due to correlated noise near the boundary. In this case, padding with a large value works better. Another common strategy which does not suffer from correlated pixels is circular padding, but its use is limited to cyclic textures. The least useful padding strategy is to repeat the border pixels, as this biases filtered values towards the border pixel values. In MATLAB, we simply choose between the three padding options:

```
xp = padarray(x, [r r], 'symmetric');
xp = padarray(x, [r r], 'circular');
xp = padarray(x, [r r],  realmax);
```

The value `realmax` stands for the largest positive floating point number. Common padding strategies can be optimized. Instead of circular padding, MATLAB's `circshift` operation can be used to avoid explicit padding. In lower-level languages like C or CUDA, we can implement padding strategies by on-the-fly calculation of indices.

9.2 Color Images

When processing color and more generally multi-spectral images, we need to consider the correlation between the channels. The channels of pixels in smooth or detail regions are more correlated than the channels of pixels in noisy regions and across edges. If the channels are not decorrelated, all the channels are filtered similarly, resulting in loss of detail.

A common pre-processing step is to transform the RGB image into a YUV color space, where the luminance Y and chromaticity components U and V are separated. The BM3D authors conclude that an opponent color transform across the channels works better than other YUV alternatives like YCbCr [52]. The opponent color transform performed on a RGB color image is essentially a 3-point *discrete cosine transform* (DCT) applied to the channels of every pixel. For hyperspectral images, a n-point DCT can be used, where n is the number of wavelengths.

In MATLAB, we can either use the `dct` function, or multiply with the DCT matrix, given by the `dctmtx` function. Using the matrix has the advantage that, for single channel grayscale images, it reduces to 1, requiring no special case handling. The DCT matrix is

$$M_{RGB\to YUV} = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{pmatrix}. \tag{9.1}$$

After processing, the image needs to be transformed back to its original color space. The MATLAB template for multi-channel based denoising looks like this:

```
[height width depth] = size(y, 3);
s = [height * width, depth];
M = dctmtx(depth)';
y = reshape(reshape(y, s) * M, size(y));
x = denoise(y);
x = reshape(reshape(x, s) / M, size(y));
```

We calculate the bilateral weights not per channel but per pixel, exactly like the the original bilateral filter [10], which sums the squared norms of the different channels. Note that this step is unaffected by the DCT, again due to its unitarity. By combining the channels, edges are reliably detected to avoid color aberration. In the frequency domain, we exploit the decorrelated channels and calculate the frequecy range kernels for every channel independently.

Besides the mentioned changes, in MATLAB we only need to replace scalar operations with `bsxfun`. The `fft2` function extends transparently to multiple channels.

9.3 Block Processing

Dual-domain filtering processes for every pixel y_p its neighborhood $y_q \in \mathcal{N}_p$ and produces an output pixel x_p. To implement this framework in MATLAB, we have at least three choices. We can explicitly loop over all pixels or use the built-in commands `nlfilter` or `blockproc`.

Manually writing the loop has the greatest flexibility, but is verbose. While `nlfilter` has a concise syntax, it still requires manual addition and removal of padding and can only handle single channel images. On the other hand `blockproc` is more complex to use, but also more powerful. It adds and removes padding, transparently handles higher dimensions, and most importantly supports parallel processing. Moreover, it allows large images to be processed on the disk. Most code examples in this thesis use `blockproc`. One example using `nlfilter` is given for artifact removal. The following three templates for block processing are equivalent and can be interchanged. Using a loop:

```
xp= padarray(x, [r r], 'symmetric');
n = zeros(size(x));
parfor p = 1:numel{x}, [i j] = ind2sub(size(x), p);
    b = xp(i:i+2*r, j:j+2*r);
    n(p) = DDF(b, ...);
end
```

Using `nlfilter`:

```
f = @(x) DDF(x, ...);
xp= padarray(x, [r r], 'symmetric');
n = nlfilter(xp, 2*[r r] + 1, f);
n = n(1+r:end-r, 1+r:end-r);
```

Using `blockproc`:

```
f = @(b) DDF(b.data, ...);
n = blockproc(x, [1 1], f, ...
  'BorderSize', [r r], 'PadMethod',   'symmetric', ...
  'TrimBorder', 0,     'UseParallel', 1);
```

9.4 FFT

MATLAB uses the *fastest Fourier transform in the west* (FFTW) as its FFT implementation. For small kernels, the overhead of the FFT outweighs the gain of the fast algorithm and standard matrix multiplication with the DFT matrix (`dftmtx`) can be faster. Alternatively, constraining the filter kernel sizes to powers of two accelerates the FFT implementation significantly.

9.4.1 Array Shifting

The standard FFT implementation of MATLAB requires shifting of the data, such that the central pixel is the first in the matrix. Unfortunately, the MATLAB documentation is unclear about the usage of the commands `fftshift` and `ifftshift`. As a consequence, many people use it wrong, or they are just lucky when they get the correct result. `fftshift` moves the origin to the center and `ifftshift` moves the center to the origin. For even array sizes, they have identical behaviour, but for odd sizes, they differ. The correct usage of the commands is: `fftshift(fft(ifftshift(x)))`. In our case, we can save the last `fftshift`, since averaging over the Fourier coefficients is order independent.

For color images and higher dimensions, we want to shift in the first two dimensions, but not in the others and `ifftshift` needs to be replaced by the more general `circshift` and `ifftshift(x)` becomes `circshift(x, -[r r])`.

9.4.2 Array Indexing

Further acceleration can be gained by using array indexing maps for reading, shifting, and flipping. We can precompute the three indexing maps as follows:

```
[dy dx] = ndgrid(-r:r);
index   = reshape(1:numel(dx), size(dx));

read    = (r+dy) + (r+dx) * (height + 2*r);
shift   = ifftshift(index);
flip    = ifftshift(rot90(fftshift(index), 2));
```

The value `height` is the height of the image in pixels. To read a patch from a padded image, the map `read` needs to be added to the linear index of the center pixel.

9.5 MATLAB Code

In this section, we present all algorithms for grayscale images as MATLAB code. For convenience, the DDF is also given in the unguided and the color variants. Adaptations of the other algorithms to handle color is straightforward by applying the template from Section 9.2.

9.5.1 Dual-Domain Filter

```
function E = DDF_unguided(y, r, a, A, k, K)

    d = y - y(1+r, 1+r);
    k = k(abs(d).^2);
    e = a * sum(sum(d .* k)) / sum(k(:));

    D = fft2(ifftshift((d - e) .* k));
    K = K(abs(D).^2 / sum(k(:).^2));
    E = A * sum(sum(D .* K)) / numel(K);
end
```

Algorithm 9.1: MATLAB code of unguided DDF for grayscale images.

```
function E = DDF(z, r, flip, a, A, k, K)

    d = z - z(1+r, 1+r);                                % (4.1)
    k = k(real(d).^2);                                  % (4.2)
    e = a * sum(sum(d .* k)) / sum(k(:));               % (4.3)

    D = fft2(ifftshift((d - e) .* k));                  % (4.4)
    K = K(abs((D + conj(D(flip)))/2).^2 / sum(k(:).^2)); % (4.5)
    E = A * sum(sum(D .* K)) / numel(K);                % (4.6)
end
```

Algorithm 9.2: MATLAB code of guided DDF for grayscale images. This version of DDF is used by the following code examples.

```
function E = DDF_color(z, r, flip, a, A, k, K)

    d = bsxfun(@minus, z, z(1+r, 1+r, :));
    k = k(sum(real(d).^2, 3));
    e = a * sum(sum(bsxfun(@times, d, k))) / sum(k(:));

    D = fft2(circshift(bsxfun(@times, ...
        bsxfun(@minus, d, e), k), -[r r]));
    K = K(abs((D + conj(D(flip)))/2).^2 / sum(k(:).^2));
    E = A * sum(sum(D .* K)) / numel(k);
end
```

Algorithm 9.3: MATLAB code of guided DDF for color images.

9.5.2 Dual-Domain Image Denoising

```
function x = DDID(y, sigma2)

    r       = 15;
    sigma_s = 7;                                                % (5.7)
    gamma_r = [100 8.7 0.7];                                    % (5.5)
    gamma_f = [4.0 0.4 0.8];                                    % (5.6)

    [dy dx] = ndgrid(-r:r);
    h       = exp(- (dx.^2 + dy.^2) / (2 * sigma_s^2));
    flip    = circshift(reshape(numel(dx):-1:1, size(dx)), [1 1]);

    x = (1 + 1i) * y;
    for i=1:3

        k = @(d2) exp(- d2 ./ (gamma_r(i) * sigma2)) .* h;      % (5.3)
        K = @(D2) 1 - exp(- gamma_f(i) * sigma2 ./ D2);         % (5.4)

        f = @(b) DDF(b.data, r, flip, 1, 1, k, K);              % (5.2)
        x = (1 + 1i) * imag(x) - imag(blockproc(x, [1 1], f, ...
            'BorderSize', [r r], 'PadMethod',   'symmetric', ...
            'TrimBorder', 0,     'UseParallel', 1));            % (5.1)
    end
    x = real(x);
end
```

Algorithm 9.4: MATLAB code of DDID.

9.5.3 Progressive Image Denoising

```
function x = PID(y, sigma2)

    N       = 30;                                                % sec 6.4
    r       = 15;
    sigma_s = 7;
    gamma_r = 988.5;
    gamma_s = 2/9;
    alpha   = 1.533;
    lambda  = log(alpha) * 0.567;
    [dy dx] = ndgrid(-r:r);
    r2      = dx.^2 + dy.^2;
    flip    = circshift(reshape(numel(dx):-1:1, size(dx)), [1 1]);

    x = y;
    for i=1:N

        T = gamma_r * sigma2    * alpha^(-i);                    % (6.6)
        S = gamma_s * sigma_s^2 * alpha^(i/2);                   % (6.7)
        k = @(d2) exp(- d2 / T) .* exp(- r2 / S);                % (6.4)
        K = @(D2) exp(- D2 / sigma2);                            % (6.5)

        f = @(b) DDF(b.data, r, flip, 0, lambda, k, K);
        x = x - real(blockproc(x, [1 1], f, ...                  % (6.3)
          'BorderSize', [r r], 'PadMethod',   'symmetric', ...
          'TrimBorder', 0,     'UseParallel', 1));
    end

    % last guided DDID step
    r       = 31;                                                % sec 6.4
    sigma_s = 16;
    gamma_r = 0.6;
    gamma_f = 2.16;
    [dy dx] = ndgrid(-r:r);
    r2      = dx.^2 + dy.^2;
    flip    = circshift(reshape(numel(dx):-1:1, size(dx)), [1 1]);

    k = @(d2) exp(- d2 / (gamma_r * sigma2)) .* exp(- r2 / (2 * sigma_s^2));
    K = @(D2) exp(- D2 / (gamma_f * sigma2));

    f = @(b) DDF(b.data, r, flip, 1, 1, k, K);
    x = y - imag(blockproc(x + 1i * y, [1 1], f, ...
      'BorderSize', [r r], 'PadMethod',   'symmetric', ...
      'TrimBorder', 0,     'UseParallel', 1));
end
```

Algorithm 9.5: MATLAB code of PID.

9.5.4 Dual-Domain Image Denoising Improved

```
function x = DDID2(y, sigma2)

    N       = 8;                                            % (7.5)
    sigma_s = 13;                                           % (7.6)
    gamma_r = 5.3 / N;                                      % (7.7)
    gamma_f = 13 / N;                                       % (7.8)
    alpha   = exp(15);                                      % (7.9)

    x = (1 + 1i) * y;
    for n = N:-1:1, t = (n - 1) / N;

        S =    2    * sigma_s^2 * alpha^(-t/2);             % (7.2)
        T = gamma_r * sigma2    * alpha^t;                  % (7.3)
        V = gamma_f * sigma2;                               % (7.4)

        r       = max(4, round(2 * sqrt(S/2)));             % (7.10)
        [dy dx] = ndgrid(-r:r);
        flip    = circshift(reshape(numel(dx):-1:1, size(dx)), [1 1]);

        a = cos(t * pi/2);                                  % (7.13)
        h = exp(- (dx.^2 + dy.^2) / S);
        k = @(d2) cos(min(pi/2, sqrt(d2/(T*n)))).^n .* h;   % (7.11)
        K = @(D2) cos(min(pi/2, sqrt(D2/(V*n)))).^n;        % (7.12)

        f = @(b) DDF(b.data, r, flip, a, a, k, K);
        x = (1 + 1i) * y - imag(blockproc(x, [1 1], f, ...  % (7.1)
          'BorderSize', [r r], 'PadMethod',   'symmetric', ...
          'TrimBorder', 0,     'UseParallel', 1));
    end
    x = real(x);
end
```

Algorithm 9.6: MATLAB code of DDID2.

9.5.5 Artifact Removal

```
function x = deart(g, y, sigma2, r, sigma_s, gamma_r, gamma_f)

    [dy dx] = ndgrid(-r:r);
    h       = exp(- (dx.^2 + dy.^2) / (2 * sigma_s^2));
    flip    = circshift(reshape(numel(h):-1:1, size(h)), [1 1]);

    k = @(d2) h .* exp(- d2 / (gamma_r * sigma2));     % (8.2)
    K = @(d2) max(0, 1 - d2 / (gamma_f * sigma2));     % (8.3)

    z = padarray(g + 1i * y, [r r], 'symmetric');
    f = @(b) DDF(b, r, flip, 1, 1, k, k);              % (8.1)
    n = nlfilter(z, size(h), f);
    x = y - imag(n(1+r:end-r, 1+r:end-r));             % (5.1)
end
```

Algorithm 9.7: MATLAB code for artifact removal.

9.6 CUDA Optimizations

To exploit the parallel structure of DDF, we implemented DDID and PID in CUDA. As a stopover, we first ported DDID first to C, using FFTW. Without the overhead of MATLAB, the execution time is halved. Multi-core parallelization was done using OpenMP and we observed linear scalability.

9.6.1 Unordered Sande-Tukey FFT

In CUDA there exists an FFT implementation called CUFFT. However, it is designed for transforming a single large array. We need for DDF a per-pixel or per-thread FFT implementation. The most popular FFT variant is the in-place radix-2 Cooley-Tukey FFT or *decimation in time* (DIT) algorithm, consisting of a reordering step followed by a butterfly. However, for our purposes, we used the Sande-Tukey-FFT [53] or *decimation in frequency* (DIF) algorithm. The DIF variant differs from DIT in that it has the reordering stage of the Fourier coefficients after the butterfly. Since we are only interested in the central pixel of the IFFT, we can calculate it by summing over all the coefficients, which is invariant with respect to the position of the coefficients. Dropping the reordering gives us a few percent performance boost. This optimization extends transparently to higher dimensions since changing the order in one dimension does not affect the other. For computing the twiddling factors, we use the faster `cospif` and `sinpif` functions. **Algorithm** 9.8 gives the resulting code.

```
__device__ void st_fft(cuComplex * f, int n)
{
    for (int m = n/2; m > 0; m >>= 1)
    for (int r =  0 ; r < n; r += m*2)
    for (int j =  0 ; j < m; j ++)
    {
        cuComplex a  = f[r + j];
        cuComplex b  = f[r + j + m];
        f[r + j]     = cuCaddf(a, b);
        f[r + j + m] = cuCmulf(cuCsubf(a, b),
            make_cuFloatComplex(cospif(-(float) j/m), sinpif(-(float) j/m)));
    }
}
```

Algorithm 9.8: CUDA code of unordered Sande-Tukey-FFT. n must be a power of two.

9.6.2 Minimize Memory Access

On GPUs, memory access is orders of magnitudes slower than arithmetic and logical operations. It is therefore important to limit reading and writing to memory, even at the cost of redundant computation. One possible GPU optimization is to apply the shift theorem to replace shifting by point-wise multiplication. This is particularly attractive, since shifting by $\frac{n}{2}$ positions in the even case reduce to flipping signs in the real and imaginary part of the Fourier coefficients.

10 Results

It doesn't matter how beautiful your theory is,
it doesn't matter how smart you are.
If it doesn't agree with experiment, it's wrong.

Richard Feynman

In this chapter, we use the implementations from Chapter 9 and evaluate them. First we analyze the robustness and optimality of the chosen parameters in Section 10.1. Then, we describe the evaluation process in Section 10.2. Section 10.3 contains the comparisons of denoised images and PSNR tables. Finally, we compare performance numbers of the denoising methods in Section 10.4.

10.1 Parameter Study

The parameter values of our algorithms were empirically found. In this section, we analyze the influence of the parameters on the denoising quality. For this analysis, we use the *Cameraman* image. Many methods specify distinct parameter sets for different noise sigma. Our parameters, on the other hand, are fixed. Since our algorithms produce good results for different noise levels, we fixed the noise sigma to $\sigma = 25$ for this analysis. For every method and parameter, we change one parameter. Except for the number of iterations N, we use either a power or a factor $0.6 \leq \beta \leq 1.4$ to perturb the parameters.

In Figure 10.1, we plot the PSNR values as functions of iteration number n or perturbation value β. Every row corresponds to a method, DDID, PID, and DDID2. The columns are the individual parameters of the methods, grouped according to correspondence across methods where applicable. With the exception of the last parameters γ_f and A, the parameters are robust against change. Figure 10.2 shows that the parameters are optimal with tolerance in the order of 0.01 dB.

The number of iterations N needs to be large enough to get good denoising results. The spatial parameters σ_s and γ_s are surprisingly unimportant. This is somewhat counter-intuitive, as the common belief is that it is connected to the noise variance. Our methods achieve good results for small and large noise ($\sigma = \{10, 25, 40\}$), without changing any parameters. The parameter controlling the range γ_r is not very important either. Again, this is counter-intuitive, as this is one of the parameters tweaked most for bilateral filtering. We suspect the robustness is due to the inverse proportional scales of the spatial and range gaussians.

The frequency range parameter γ_f of DDID and DDID2 and the gradient descent factor A of PID have large influence on the PSNR. This is not surprising, since they have strong correlation with the noise estimation. Small values mean that the noise will be underestimated and consequently the image will contain residual noise. Conversely, large values mean that the noise is overestimated and the image will loose details.

The plots also confirm our experience, that these parameters were the ones required tuning the most. We show in Table 10.1 the resulting images when the frequency range parameter γ_f or the gradient descent factor A is changed.

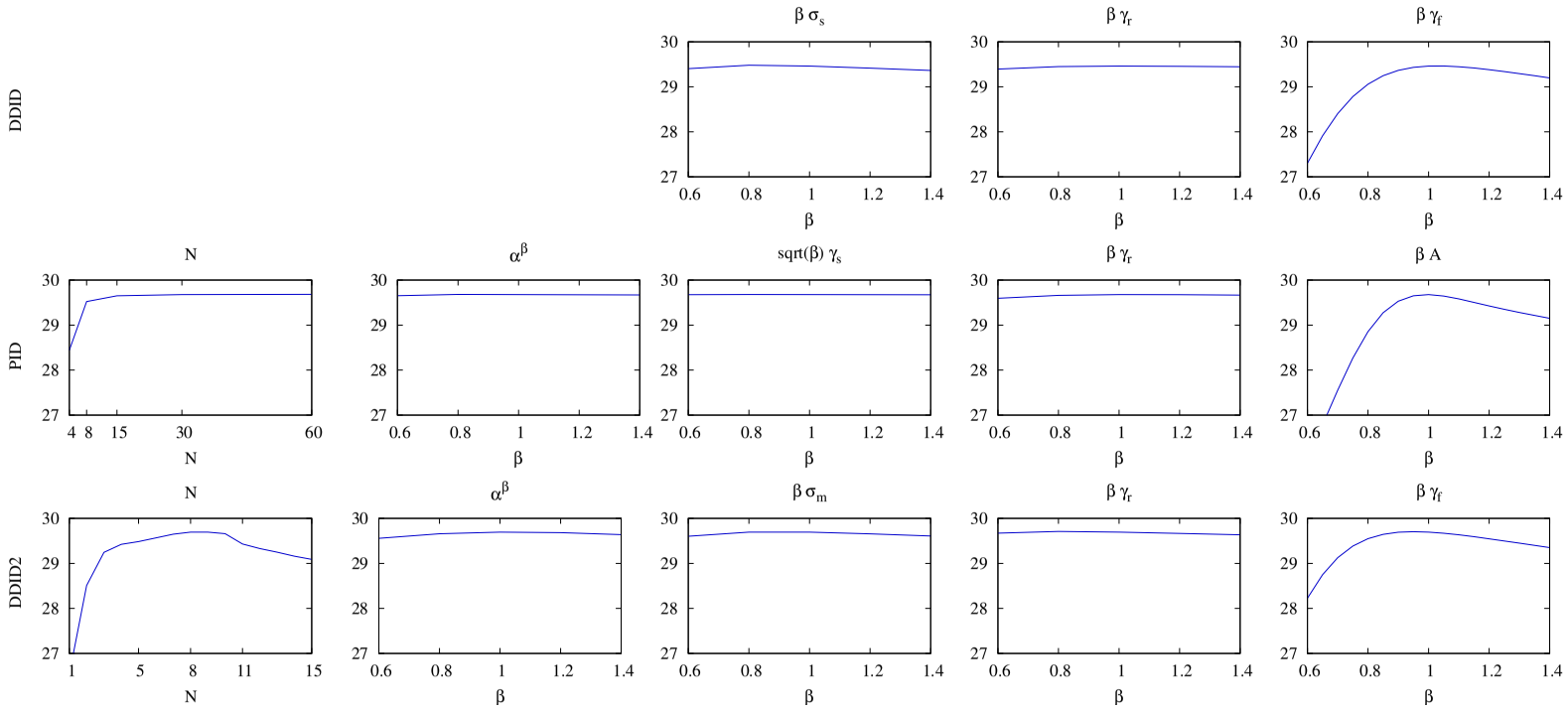


Figure 10.1: *Plots of* PSNR *(dB) against parameter change at* $10\times$ *finer scale. Our algorithms are robust with respect to changes of most parameters. The most influential parameters are* γ_f *and* A*, which control the noise estimation directly.*

10.2 Evaluation Process

In this section, we describe our evaluation method. For image denoising, we evaluate the denoising quality with a threefold loop over every image, noise sigma, and method. In every iteration, we load the original image x, add white Gaussian noise n with the noise sigma σ to it, denoise using the method, save the image $\hat{x}$ and calculate the PSNR value. The saved images are either compared directly, or derived images like differences of the images and difference of the squared errors are used for visualization.

For artifact removal, we use the denoised image as the guide g to DDF and the noisy image y as input. For JPEG deblocking, we have only a single image, so we have $g = y$, and we feed this image to DDF.

10.2.1 Benchmark Images

Most of the natural images are from the BM3D test images. They are shown in Figure 10.3 and Figure 10.4. We also used synthetic images as shown in Figure 10.5.

We used the following grayscale images

- SIPI images [54]: *Barabara, Boats, Cameraman, Couple, Finger Print, Hill, House, Lena, Man, Pepper*
- BM3D [55]: *Montage*
- CSF chart (see Section 10.2.2): CSF *soft*, CSF *hard*,

and the following RGB color images for benchmark

- SIPI images [54]: *Baboon, F-16, House, Lake, Lena, Pepper, Tiffany*
- Kodak: *Kodak 1, Kodak 2, Kodak 3, Kodak 12*
- Artworks by Charis Tsevis [56]: *Nikon, Basketball*
- Logo Open Stock [57]: *Organics*.

10.2.2 CSF Chart

We use the CSF chart function described by Pelli [58].

$$f(x, y) = \frac{1}{2}\left(1 + 10^{-y} \sin 10^{x}\right) \tag{10.1}$$

We bound the signal bandwidth by Nyquist frequency by constraining $x \in \left[0, \frac{W(\pi w)}{\log 10}\right]$, where $W(\cdot)$ is the Lambert W function and w the width of the image

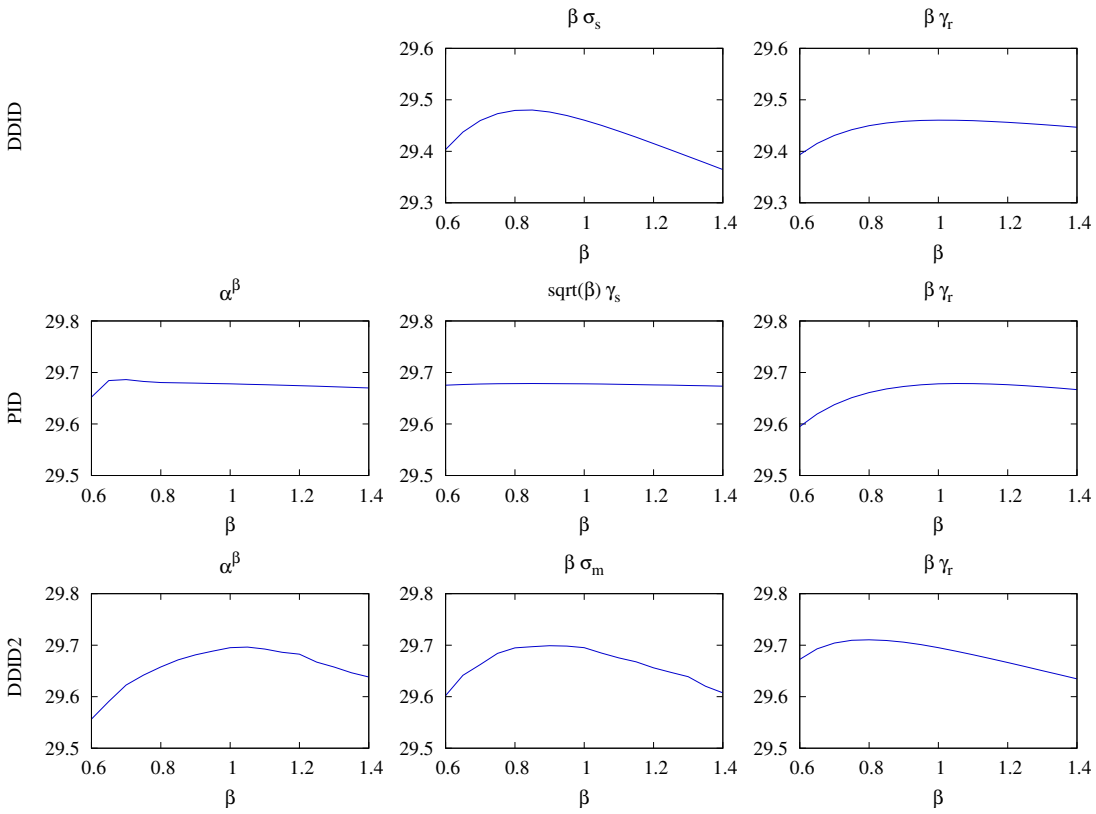
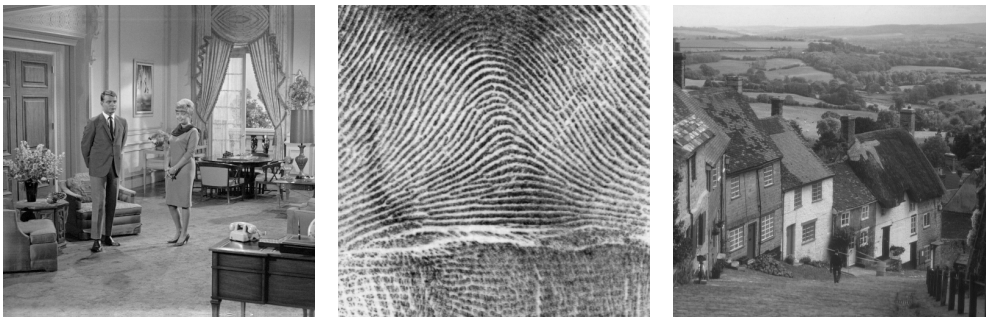


Figure 10.2: *Plots of* PSNR *(dB) against parameter change. The values chosen for the less influential parameters are close to the optimal value. Changing them has little impact on the* PSNR*, in the order of 0.1 dB.*

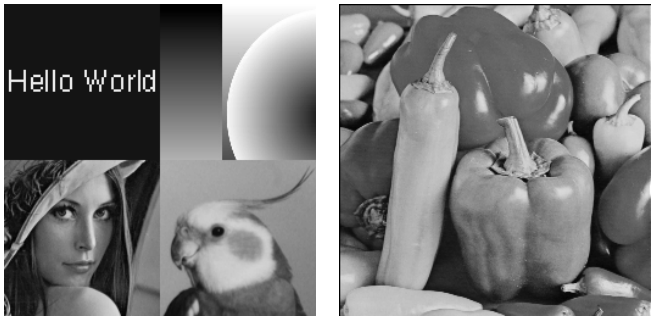
Table 10.1: *The images with small frequency domain parameters are too noisy, the images with large frequency domain parameters are too blurry.*

Barbara Boats Cameraman



Couple Finger Print Hill

House Lena Man



Montage Pepper

Figure 10.3: *The grayscale* BM3D *test images are from* SIPI*. The only exception is* Montage*, which is from the authors of* BM3D *and contains synthetic elements.*

Baboon

F-16

House

Kodak 1

Kodak 2

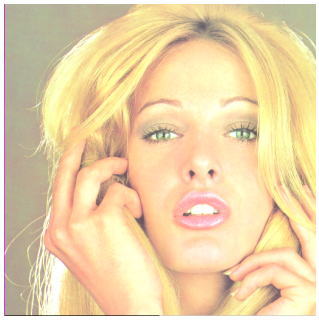
Kodak 3

Kodak 12

Lake

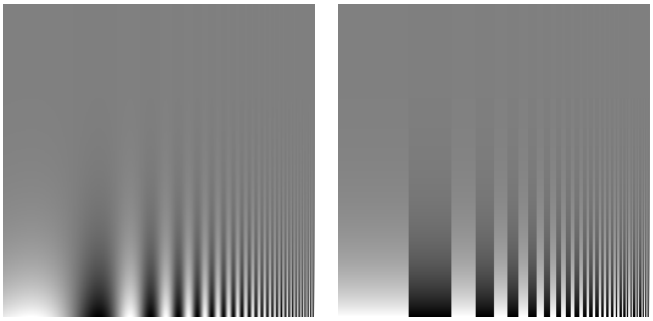
Lena

Pepper



Tiffany

Figure 10.4: *The* BM3D *test images are mostly from* SIPI. *The others are from Kodak.*



CSF soft CSF hard

Organics Nikon Basketball

Figure 10.5: *Synthetic images for benchmark in grayscale and color.* Organics *is from Logo Open Stock.* Nikon *and* Basketball *are from the artist Charis Tsevis.* CSF *soft and* CSF *hard we created using Pulli's formula.*

in pixels. We allow the contrast to decrease to 10^{-3} with $y \in [0,3]$. For our version with steps, we use

$$f(x,y) = \frac{1}{2}\left(1 + 10^{-y}\,\mathrm{sgn}\left(\sin 10^{x}\right)\right). \tag{10.2}$$

Algorithm 10.1 gives a MATLAB implementation for producing CSF charts.

```
function image = CSF(w, h, hard)

    s = lambertw(pi * w) / log(10);

    [x y] = meshgrid(linspace(0, s, w), linspace(3, 0, h));

    if hard, image = 0.5 * (1 + (10.^(-y) .* sign(sin(10.^x))));
    else     image = 0.5 * (1 + (10.^(-y) .*      sin(10.^x)));
    end
end
```

Algorithm 10.1: MATLAB code for generating CSF charts. The parameters w and h define the image size and the parameter *hard* chooses between sinusoidal and rectangular waves.

10.2.3 Noise Generation

For grayscale images, we choose $\sigma \in \{10, 25, 40\}$. For color images, we choose $\sigma \in \{25, 40\}$. Note that the noise sigma is given assuming the image is scaled between 0 and 255. We use the same scale for the noise sigma, as it is common in the literature.

We use the seed value 0 to initialize the normally distributed random value generator `randn` of MATLAB used to generate the white Gaussian noise:

```
randn('seed', 0);
n = randn(size(x)) * sigma;
y = x + n;
```

Exceptions are the algorithms that are implemented in pure C (NLB, NLM, TV), as the noise is generated differently.

10.2.4 Image Denoising Methods

We run our algorithms against the state-of-the-art methods listed in Table 10.2. Most of the implementations can be found on Xin Li's website on *reproducible research in computational science* [77].

10.3 Images and Tables

In this section, we first compare the denoising progress of our proposed methods. Then, we present comparisons between our methods and other state-of-the-art methods.

Figure 10.6, Figure 10.7, and Figure 10.8 give visual comparison of the denoising progress for DDID, PID, and DDID2. Figure 10.9 plots the PSNR and MSE curves over the denoising progress for *Cameraman*. The curves for denoising other images look similar. Both the image series and the plots suggest that relatively, PID denoises faster and more steadily than DDID2.

Figure 7.3 and Figure 7.4 compare denoising results of a natural image. Figure 7.5 and Figure 7.6 compare denoising results of a synthetic image. In both cases, PID and DDID2 produce numerically and visually the best results.

Finally, Table 10.3, Table 10.4, and Table 10.5 compare the PSNR values of the images denoised by the various methods. Our methods are competitive for denoising grayscale images and have the best PSNR values for color images. Figure 10.10 gives a graphical representation of the tables.

Name	Paper	Code	Color	Language
DDID2	[4]		yes	MATLAB
PID	[3]		yes	MATLAB, CUDA
DDID	[2]	[59]	yes	MATLAB, C, CUDA
BM3D	[26]	[55]	yes	MATLAB + C
BM3D-SAPCA	[31]	[55]	no	MATLAB
FOE	[60]	[61]	yes	MATLAB
ISKR	[19]	[62]	no	MATLAB
K-LLD	[50]	[63]	no	MATLAB
K-SVD	[33, 64]	[65, 66]	yes	MATLAB, C
LSSC	[34]	[67]	no	MATLAB + C
MLP	[35]	[68]	no	MATLAB
NLB	[27, 69]	[70]	yes	C
NLM	[20, 71]	[72]	yes	C
PLOW	[28]	[73]	yes	MATLAB + C
SAIST	[29]	[74]	no	MATLAB
TV	[18, 75]	[76]	yes	C

Table 10.2: *These are the implementations of image denoising methods we considered for comparison.*

Figure 10.6: *Denoising progress of* DDID. *The noise sigma is* $\sigma = 40$.

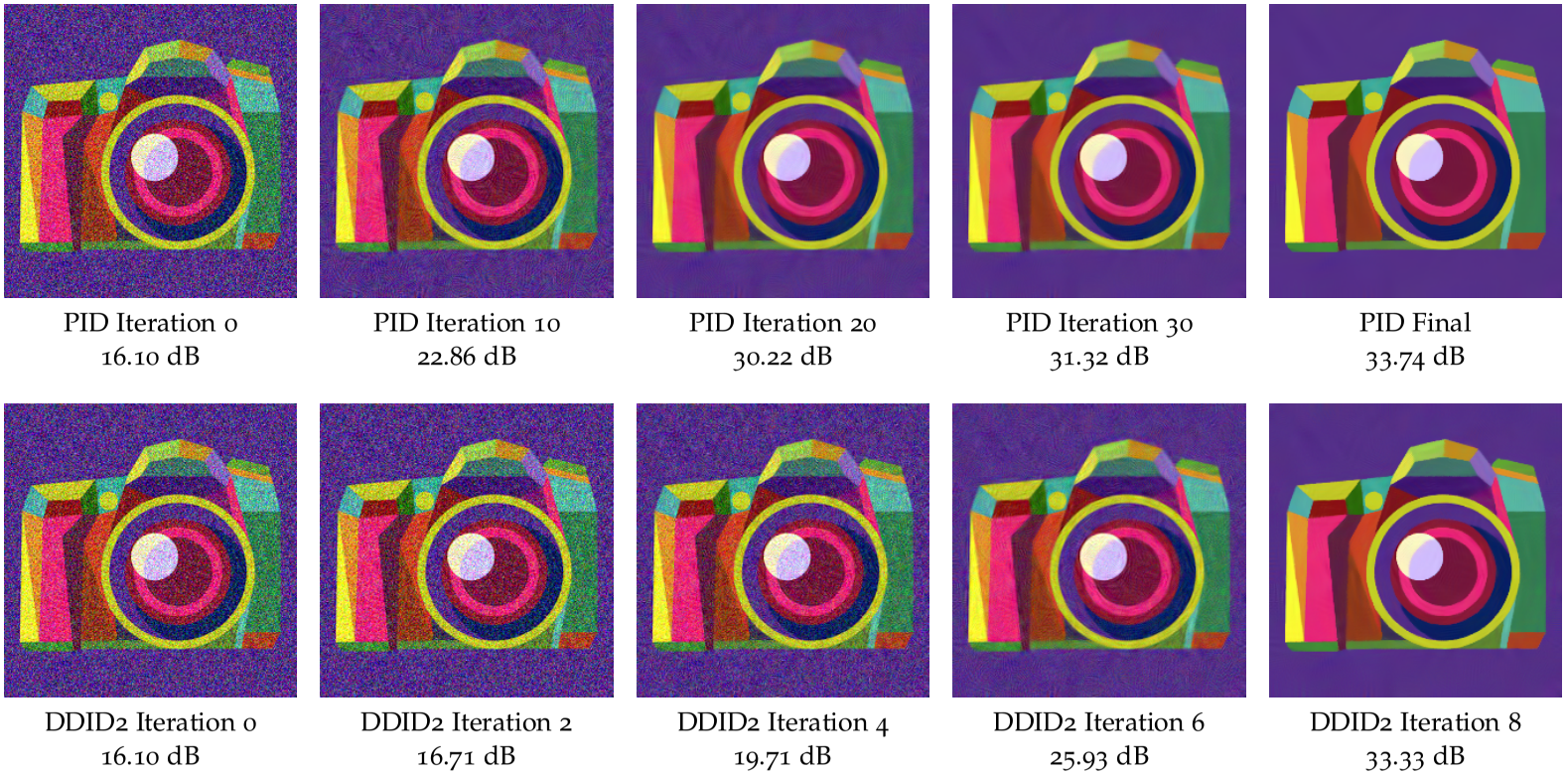


Figure 10.7: *denoising progress of* PID *and* DDID2*. The noise sigma is* $\sigma = 40$*. The image* PID Iteration 30 *is used as an oracle to produce the final image with a single* DDID *step.*

Figure 10.8: *Denoising progress of* PID *and* DDID2. *The noise sigma is* $\sigma = 40$. *The image* PID Iteration 30 *is used as an oracle to produce the final image with a single* DDID *step.*

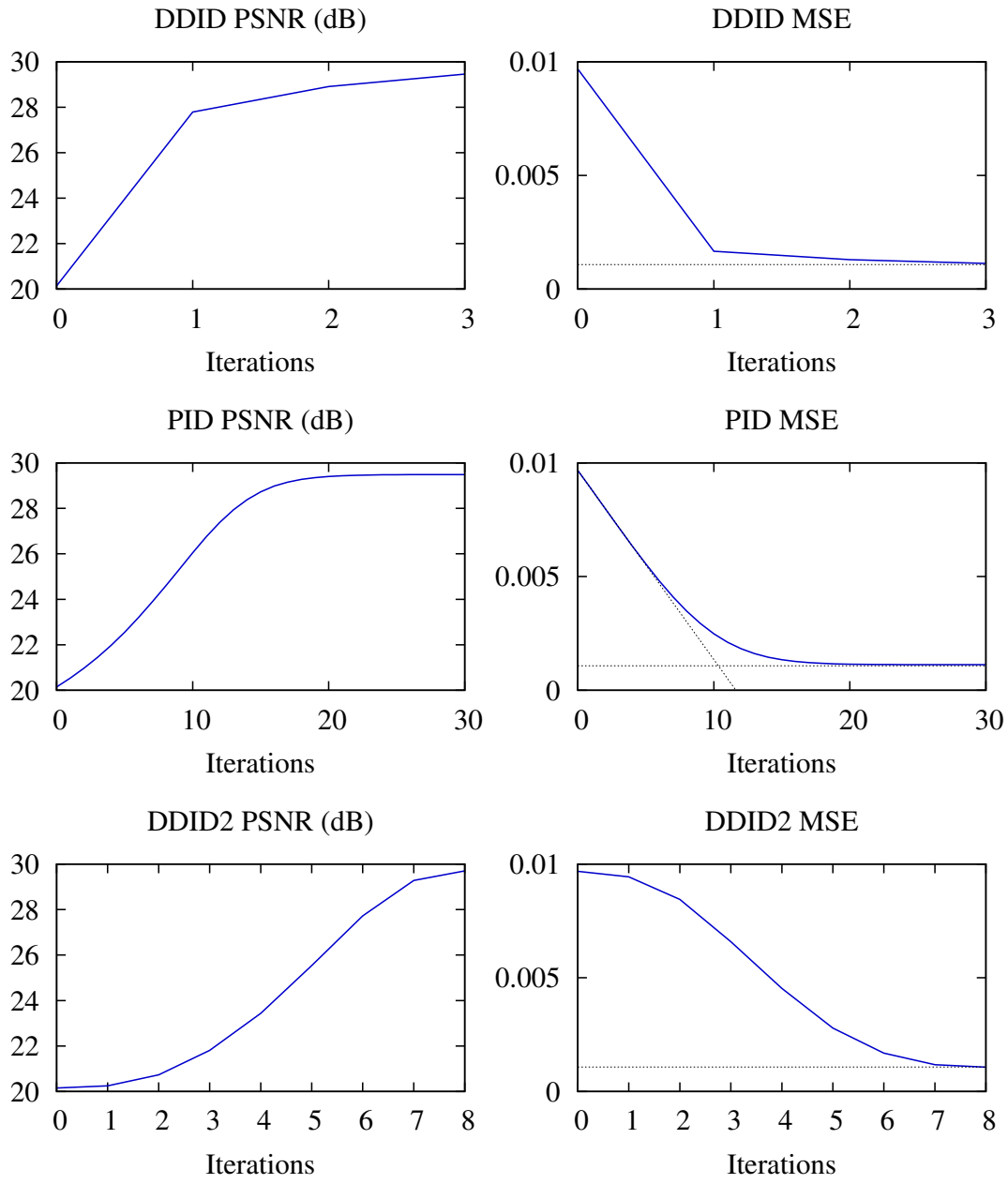


Figure 10.9: *Quality improvement over iterations for cameraman. The curves for* PID *and* DDID2 *are qualitatively different.*

Grayscale	DDID2	PID	DDID	SAPCA	SAIST	BM3D	MLP	LSSC	NLB	K-SVD	NLM	PLOW	KLLD	FOE	TV	ISKR
Barbara	34.66	34.55	34.67	35.10	**35.19**	34.98	34.07	34.99	34.44	34.44	33.18	33.80	33.13	32.87	29.80	27.99
Boats	33.83	33.77	33.74	**34.10**	33.93	33.92	33.81	34.03	33.57	33.65	32.89	32.98	33.01	33.09	31.24	27.53
Cameraman	34.21	34.14	34.05	**34.59**	34.23	34.18	34.18	34.24	34.04	33.76	33.43	33.17	32.82	33.59	31.20	26.01
Couple	33.98	33.91	33.88	**34.17**	34.00	34.04	33.91	34.01	33.69	33.55	32.88	33.13	33.11	33.25	30.97	27.99
Finger Print	31.85	31.78	31.84	32.64	**32.68**	32.46	32.57	32.57	32.10	32.40	31.00	31.04	31.66	32.03	29.14	25.99
Hill	33.65	33.60	33.56	**33.83**	33.70	33.62	33.59	33.67	33.52	33.37	32.86	32.63	32.79	32.58	31.38	27.99
House	36.66	36.60	36.50	**37.01**	36.81	36.71	35.98	36.95	35.93	35.97	34.95	36.24	35.26	35.12	33.17	29.88
Lena	35.92	35.81	35.81	**36.07**	35.85	35.93	35.85	35.85	35.36	35.50	34.34	35.32	35.27	35.04	32.81	30.73
Man	34.12	34.03	34.02	**34.25**	34.13	33.98	34.11	34.10	33.88	33.60	33.09	32.96	33.19	33.18	31.53	27.35
Montage	37.71	37.43	37.51	**37.85**	37.19	37.35	36.51	37.26	36.91	36.10	35.32	36.14	35.44	36.26	33.14	27.72
Pepper	34.75	34.67	34.58	**34.94**	34.79	34.68	34.72	34.80	34.46	34.25	33.50	33.57	33.86	34.25	31.94	26.26

$\sigma = 10$

Table 10.3: PSNR *(dB) values for denoising grayscale images with noise sigma* $\sigma = 10$. DDID2 *can compete with the best grayscale denoisers* MLP, BM3D, BM3D-SAPCA, *and* SAIST.

Grayscale	DDID2	PID	DDID	SAPCA	SAIST	BM3D	MLP	LSSC	NLB	K-SVD	NLM	PLOW	KLLD	FOE	TV	ISKR
Barbara	30.82	30.56	30.80	31.00	**31.23**	30.72	29.55	30.49	30.13	29.57	28.98	30.21	27.69	26.25	24.91	28.26
Boats	29.88	29.80	29.79	**30.03**	29.97	29.91	29.97	29.90	29.54	29.32	28.56	29.54	29.26	27.68	27.05	28.56
Cameraman	29.69	29.68	29.47	**29.81**	29.41	29.45	29.61	29.50	29.35	28.91	28.76	28.66	28.42	26.76	26.09	27.12
Couple	29.67	29.65	29.56	**29.82**	29.74	29.72	29.74	29.67	29.24	28.89	28.26	29.36	29.15	27.49	26.63	28.42
Finger Print	27.34	27.15	27.32	27.81	**27.93**	27.70	27.65	27.62	27.47	27.26	26.17	27.05	26.97	25.09	23.84	26.05
Hill	29.80	29.77	29.71	**29.96**	29.90	29.85	29.88	29.83	29.55	29.23	28.63	29.61	29.25	27.94	27.66	28.63
House	32.90	32.84	32.66	32.96	**33.17**	32.86	32.57	33.13	32.26	32.08	31.29	32.73	31.49	28.83	29.49	31.20
Lena	**32.27**	32.12	32.14	32.23	32.26	32.08	32.26	31.86	31.66	31.36	30.47	31.91	31.43	29.37	29.05	31.32
Man	29.71	29.68	29.62	29.81	29.75	29.62	**29.89**	29.70	29.52	29.11	28.58	29.34	29.28	27.74	27.60	28.46
Montage	**33.08**	32.76	32.61	32.97	32.35	32.37	32.04	32.25	31.77	31.16	30.63	30.96	30.48	28.43	27.08	28.76
Pepper	**30.46**	30.37	30.29	30.43	30.40	30.16	30.31	30.23	30.01	29.71	28.83	29.64	29.56	27.49	26.99	28.10

$\sigma = 25$

Grayscale	DDID2	PID	DDID	SAPCA	SAIST	BM3D	MLP	LSSC	NLB	K-SVD	NLM	PLOW	KLLD	FOE	TV	ISKR
Barbara	28.59	28.38	28.51	**28.68**	28.62	27.99	n/a	28.17	27.96	26.89	26.72	28.10	24.84	18.98	23.00	22.98
Boats	27.75	27.71	27.65	**27.92**	27.62	27.74	n/a	27.77	27.35	27.06	26.22	27.63	26.33	19.18	24.70	23.46
Cameraman	27.55	**27.60**	27.32	27.57	27.29	27.18	n/a	27.34	27.18	26.76	26.51	26.66	25.57	19.20	23.13	22.98
Couple	27.39	27.40	27.30	**27.58**	27.33	27.48	n/a	27.41	27.03	26.39	25.70	27.33	26.19	19.15	24.40	23.23
Finger Print	25.15	24.98	25.04	25.54	**25.55**	25.30	n/a	25.30	25.36	24.70	24.11	25.21	24.00	18.82	20.87	20.77
Hill	27.93	27.92	27.83	**28.08**	27.87	27.99	n/a	28.00	27.67	27.21	26.43	27.89	26.54	19.21	25.52	23.96
House	30.63	30.76	30.41	30.75	**31.38**	30.65	n/a	31.10	30.27	29.59	28.80	30.55	27.43	19.07	26.82	24.38
Lena	**30.22**	30.14	30.07	30.10	30.03	29.86	n/a	29.91	29.70	29.05	28.16	29.86	27.64	19.27	26.71	24.89
Man	27.67	27.66	27.60	**27.83**	27.58	27.65	n/a	27.64	27.41	27.05	26.30	27.52	26.42	19.20	25.45	23.81
Montage	**30.26**	30.25	29.82	30.02	29.50	29.52	n/a	29.43	28.97	28.63	27.44	27.90	26.62	19.50	22.80	24.59
Pepper	**28.10**	**28.10**	27.94	**28.10**	27.94	27.70	n/a	27.86	27.56	27.40	25.65	27.50	26.15	19.08	23.79	23.38

$\sigma = 40$

Table 10.4: PSNR *(dB) values for denoising grayscale images.* DDID2 *can compete with the best grayscale denoisers* MLP, BM3D, BM3D-SAPCA, *and* SAIST.

Color	DDID2	PID	DDID	BM3D	NLB	K-SVD	NLM	PLOW
Baboon	26.29	26.12	26.17	25.95	**26.50**	25.86	25.63	24.22
F-16	**33.06**	33.02	32.88	32.78	32.91	32.32	31.33	30.98
House	32.88	32.90	32.69	**33.03**	32.59	32.08	31.47	31.92
Kodak 1	**29.29**	29.18	29.09	29.13	29.25	28.88	27.55	25.68
Kodak 2	**32.49**	32.40	32.29	32.44	32.25	31.97	30.72	29.86
Kodak 3	**34.72**	34.70	34.55	34.54	34.43	33.20	32.43	30.46
Kodak 12	33.64	33.55	33.46	**33.76**	33.39	32.72	31.68	30.00
Lake	28.99	28.93	28.85	28.68	**29.05**	28.73	28.28	27.60
Lena	**32.45**	32.41	32.30	32.27	32.19	31.69	30.88	31.01
Pepper	**31.40**	31.36	31.25	31.20	31.19	30.74	30.32	30.35
Tiffany	**32.65**	32.61	32.49	32.23	32.17	31.66	31.12	31.42

$\sigma = 25$

Color	DDID2	PID	DDID	BM3D	NLB	K-SVD	NLM	PLOW
Baboon	24.33	24.29	24.19	23.87	**24.45**	23.79	23.22	22.56
F-16	31.07	**31.09**	30.84	30.25	30.77	29.78	28.65	29.05
House	31.25	**31.35**	30.93	30.60	30.82	29.94	29.15	30.05
Kodak 1	**26.97**	26.91	26.77	26.59	26.79	26.45	24.79	24.32
Kodak 2	**30.66**	30.60	30.46	30.30	30.25	29.69	28.39	28.22
Kodak 3	32.42	**32.46**	32.22	31.57	32.03	30.55	30.01	28.45
Kodak 12	**31.69**	31.65	31.46	31.31	31.31	30.40	29.46	28.02
Lake	**27.39**	27.36	27.23	26.88	27.35	26.92	26.03	26.02
Lena	**30.87**	**30.87**	30.68	30.11	30.42	29.66	28.87	29.26
Pepper	**30.05**	30.04	29.88	29.27	29.63	29.05	28.23	28.68
Tiffany	30.93	**30.95**	30.73	30.13	30.25	29.81	28.76	29.73

$\sigma = 40$

Table 10.5: PSNR *(dB) values for denoising color images.* DDID2 *and* PID *have the highest* PSNR *values.*

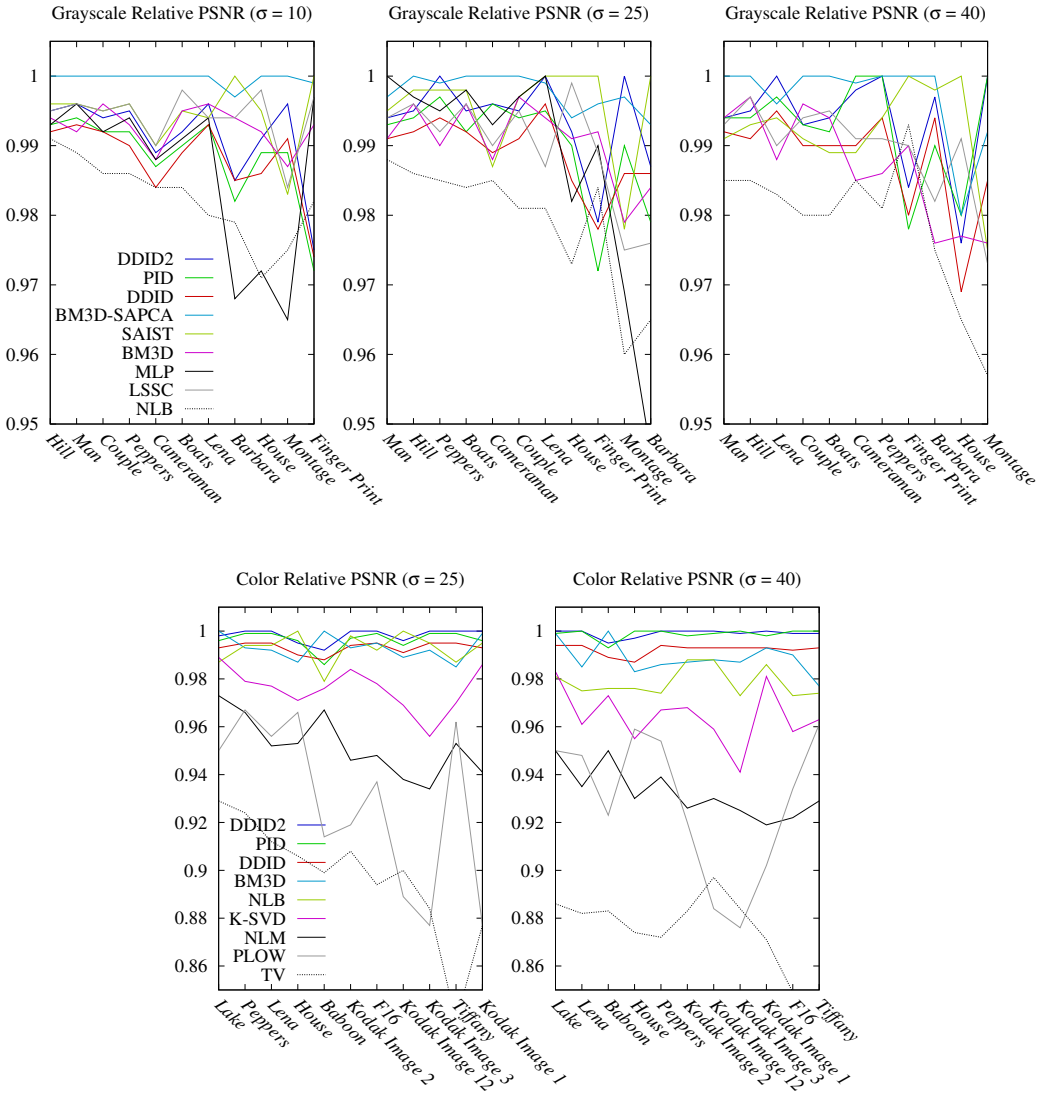


Figure 10.10: *Graphical comparison of image denoising methods. In Y-direction, we measure the relative* PSNR *values compared to the best* PSNR *value obtained considering all denoising methods. In X-direction, we order the images according to increasing variance in relative* PSNR *values. For grayscale images, we compare the best 9 methods, and for color images, we compare all methods supporting color. For grayscale images with low noise-contamination,* BM3D-SAPCA *has the best numerical results. For higher noise-contamination,* PID *and* DDID2 *can beat* BM3D-SAPCA *for some images. For color images, especially with high noise-contamination,* PID *and* DDID2 *perform best.*

10.4 Performance

Here we present a performance comparison between different methods. It is difficult to have a fair comparison, as some implementations are written in MATLAB and some in native C. We therefore separated the results into two categories, the first uses only MATLAB, while the second may also use native C code. We also include numbers for our CUDA implementations of DDID and PID. For MATLAB implementations, we restricted the use of cores by calling `matlabpool close` and `maxNumCompThreads(1)`. For native implementations, we called `taskset -c 0` to bind a process to a single core. The test system has dual quad-core Intel Xeon 2.67 GHz CPUs and runs Linux. The GPU is an NVIDIA Titan.

For benchmark, we denoise *Cameraman*, which has an image size of 256×256 pixels. The noise sigma is $\sigma = 25$. The measurements for pure MATLAB implementations are shown in Table 10.6. The results for native implementations are shown in Table 10.7.

Method	1 Core	8 Cores
DDID2	281	41
PID	814	113
DDID	95	14
BM3D-SAPCA	219	
FOE	215	
ISKR	207	
K-LLD	663	
K-SVD	249	
MLP	27	
SAIST	474	

Table 10.6: *Speed comparison of* MATLAB *implementations in seconds. Our algorithms are easy to parallelize and scale almost linearly.*

Method	1 Core	8 Cores	GPU
PID			<3
DDID	40	5	1
BM3D	<2		
LSSC	1546		
NLB	<2		
NLM	5		
PLOW	873		

Table 10.7: *Speed comparison of native implementations in seconds. Our algorithms are easy to port to* GPU *and we observe great speedup.*

11 Conclusions

You wanna know how I did it?
I never saved anything for the swim back.

Vincent, Gattaca

In this work, we have shown that significant progress can be made in a classic field. Our methodology has led us to the discovery of a new class of image denoising methods which produce high-quality results. Our image denoising methods PID and DDID2 are highly competitive for natural and grayscale images with state-of-the-art algorithms. For synthetic images and color images, our methods exceed and redefine the state-of-the-art.

Our algorithms are surprisingly simple thanks to the discovery of the DDF, which is not based on patches. The DDF complements bilateral filtering with frequency domain filtering and gives control over the filtering process in two domains. In both domains, robust noise estimators can be freely chosen. We have demonstrated that the DDF can stand on its own feet for removal of artifacts like residual noise and compression artifacts. The simplicity and quality of DDF suggest that it has the potential to become a universal tool for image enhancement and restoration like the bilateral filter.

The DDF has allowed us to gain two key insights about the image denoising process. First, denoising should be considered as robust noise estimation in multiple domains. This is a departure from the traditional denoising model, where the original image is estimated directly. The common wavelet based filtering paradigm is limited to three basic kernel functions. With DDF, the robust noise estimation model invites exploration of numerous estimators from robust statistics. Second, denoising is an iterated physical process. PID and DDID2 have demonstrated that the scales and shapes of the robust kernels smoothly change over time. With these insights, we are at least two steps closer to understand what denoising is. The DDF can serve as a primitive for further

experiments, allowing us to expect even better understanding of the denoising process.

11.1 Future

With our final words, we risk an outlook into the future. Better image denoising quality impacts any image processing system, like it is found in digital cameras. Today, state-of-the-art denoising methods are still too slow for integration into consumer products. Digital images are getting larger by the mega-pixels and consumers demand interactive performance, yet many methods take minutes to process even a small test image. Simpler methods have the advantage that analytical tricks can lead to performance improvements. The introduction of the bilateral filter to image processing has spurred many innovative works, successfully accelerating the method [13, 78, 79]. For DDF, we hope for a repetition of this phenomenon.

The universality of denoising algorithms promises adaptations to many problems by small modifications. For example, a small change in Equation 4.8 makes our DDF formulation become agnostic of dimensionality of the signal. We expect therefore that our denoising and deartifact algorithms are also applicable to volumetric datasets and video. We also expect that adaptations can be made for cases where the AWGN assumptions do not hold. For example, Rousselle et al. have demonstrated the adaptation of NLM to non-homogeneous noise with unknown distributions [80].

Image denoising is a starting point for many related problems. Super-resolution, inpainting, deblurring, demosaicking, detail enhancement, and segmentation are related to image denoising.

While our research has been empirical, eventually the goal is to have a better theoretical handle to improve parametrization of DDF and our denoising algorithms. For this, we ask for the helpful response of the community.

References

[1] F. J. Anscombe, "The transformation of poisson, binomial and negative-binomial data," *Biometrika*, vol. 35, no. 3/4, pp. 246–254, 1948.

[2] C. Knaus and M. Zwicker, "Dual-domain image denoising," in *Proceedings of the IEEE International Conference on Image Processing*. IEEE, 2013.

[3] ———, "Progressive image denoising," *Image Processing, IEEE Transactions on (under review)*.

[4] ———, "Dual-domain filtering," 2014.

[5] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, 2004.

[6] A. Horé and D. Ziou, "Is there a relationship between peak-signal-to-noise ratio and structural similarity index measure?" *Image Processing, IET*, vol. 7, no. 1, pp. 12–24, 2013.

[7] R. Mantiuk, K. J. Kim, A. G. Rempel, and W. Heidrich, "Hdr-vdp-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions," *ACM Trans. Graph.*, vol. 30, no. 4, p. 40, 2011.

[8] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 7, pp. 629–639, 1990.

[9] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger, "Robust anisotropic diffusion," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 421–432, 1998.

[10] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998, pp. 839–846.

[11] M. Elad, "On the origin of the bilateral filter and ways to improve it," *Image Processing, IEEE Transactions on*, vol. 11, no. 10, pp. 1141–1151, 2002.

[12] D. Barash, "A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 6, pp. 844–847, 2002.

[13] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 257–266, 2002.

[14] S. Paris, P. Kornprobst, and J. Tumblin, *Bilateral filtering: Theory and applications*. Now Publishers Inc, 2009, vol. 1.

[15] J. Wang, D. Xu, C. Lang, and B. Li, "An adaptive tone mapping method for displaying high dynamic range images." *J. Inf. Sci. Eng.*, vol. 26, no. 3, pp. 977–990, 2010.

[16] B. Zhang and J. P. Allebach, "Adaptive bilateral filter for sharpness enhancement and noise removal," *Image Processing, IEEE Transactions on*, vol. 17, no. 5, pp. 664–678, 2008.

[17] M. Zhang and B. K. Gunturk, "Compression artifact reduction with adaptive bilateral filtering," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2009, pp. 72 571A–72 571A.

[18] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

[19] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *Image Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 349–366, 2007.

[20] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[21] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[22] D. L. Donoho, "De-noising by soft-thresholding," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 613–627, 1995.

[23] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *Image Processing, IEEE Transactions on*, vol. 12, no. 11, pp. 1338–1351, 2003.

[24] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the american statistical association*, vol. 90, no. 432, pp. 1200–1224, 1995.

[25] A. Levin, B. Nadler, F. Durand, and W. T. Freeman, "Patch complexity, finite pixel correlations and optimal denoising," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 73–86.

[26] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. on Image Process.*, vol. 16, no. 8, pp. 2080 –2095, aug. 2007.

[27] M. Lebrun, A. Buades, and J. Morel, "A nonlocal bayesian image denoising algorithm," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1665–1688, 2013.

[28] P. Chatterjee and P. Milanfar, "Patch-based near-optimal image denoising," *Image Processing, IEEE Transactions on*, vol. 21, no. 4, pp. 1635–1649, 2012.

[29] X. Li, W. Dong, and G. Shi, "Nonlocal image restoration with bilateral variance estimation: a low-rank approach," 2013.

[30] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "A non-local and shape-adaptive transform-domain collaborative filtering," in *Proc. 2008 Int. Workshop on Local and Non-Local Approximation in Image Processing, LNLA 2008*, 2008.

[31] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian *et al.*, "Bm3d image denoising with shape-adaptive principal component analysis," in *SPARS'09*, 2009.

[32] J. F. Murray and K. Kreutz-Delgado, "Learning sparse overcomplete codes for images," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 45, no. 1-2, pp. 97–110, 2006.

[33] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[34] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2272–2279.

[35] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?" in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2392–2399.

[36] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 2833–2840.

[37] P. Chatterjee and P. Milanfar, "Is denoising dead?" *Image Processing, IEEE Transactions on*, vol. 19, no. 4, pp. 895–911, 2010.

[38] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3736–3745, 2006.

[39] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," in *ACM Trans. Graph.*, vol. 23. ACM, 2004, pp. 664–672.

[40] V. Nath, D. Hazarika, and A. Mahanta, "Blocking artifacts reduction using adaptive bilateral filtering," in *Signal Processing and Communications (SPCOM), 2010 International Conference on*. IEEE, 2010, pp. 1–5.

[41] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive dct for high-quality deblocking of compressed color images," in *14th Eur. Signal Process. Conf., EUSIPCO*, 2006.

[42] K. He, J. Sun, and X. Tang, "Guided image filtering," *Computer Vision–ECCV 2010*, pp. 1–14, 2010.

[43] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes in c: The art of scientific computing," 1992.

[44] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital image processing using MATLAB*. Gatesmark Publishing Knoxville, 2009, vol. 2.

[45] M. Lebrun, "An Analysis and Implementation of the BM3D Image Denoising Method," *Image Processing On Line*, vol. 2012, 2012.

[46] H. Yu, L. Zhao, and H. Wang, "Image denoising using trivariate shrinkage filter in the wavelet domain and joint bilateral filter in the spatial domain," *IEEE Trans. on Image Process.*, vol. 18, no. 10, pp. 2364 –2369, oct. 2009.

[47] S. Albrecht, I. Cumming, and J. Dudas, "The momentary fourier transformation derived from recursive matrix transformations," in *Digital Signal Processing Proceedings, 1997. DSP 97., 1997 13th International Conference on*, vol. 1. IEEE, 1997, pp. 337–340.

[48] S. Z. Li, "Robustizing robust m-estimation using deterministic annealing," *Pattern Recognition*, vol. 29, no. 1, pp. 159–166, 1996.

[49] R. Frühwirth and W. Waltenberger, "Redescending m-estimators and deterministic annealing, with applications to robust regression and tail index estimation," *arXiv preprint arXiv:1006.3707*, 2010.

[50] P. Chatterjee and P. Milanfar, "Clustering-based denoising with locally learned dictionaries," *Image Processing, IEEE Transactions on*, vol. 18, no. 7, pp. 1438–1451, 2009.

[51] W. Dong, G. Shi, and X. Li, "Nonlocal image restoration with bilateral variance estimation: A low-rank approach," *Image Processing, IEEE Transactions on*, vol. 22, no. 2, pp. 700–711, 2013.

[52] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 1. IEEE, 2007, pp. I–313.

[53] W. M. Gentleman and G. Sande, "Fast fourier transforms: for fun and profit," in *Proceedings of the November 7-10, 1966, fall joint computer conference*. ACM, 1966, pp. 563–578.

[54] Signal and image processing institute. [Online]. Available: sipi.usc.edu/database/

[55] Image and video denoising by sparse 3d transform-domain collaborative filtering. [Online]. Available: www.cs.tut.fi/~foi/GCF-BM3D/

[56] Charis tsevis. [Online]. Available: www.tsevis.com

[57] Logo open stock. [Online]. Available: www.logoopenstock.com

[58] D. G. Pelli, "Programming in postscript: Imaging on paper from a mathematical description." *BYTE*, vol. 12, no. 5, pp. 185–202, 1987.

[59] Claude knaus. [Online]. Available: http://knuth.unibe.ch/~knaus/

[60] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 860–867.

[61] Stefan roth. [Online]. Available: http://www.gris.informatik.tu-darmstadt.de/~sroth/research/software.html

[62] Hiroyuki takeda. [Online]. Available: http://users.soe.ucsc.edu/~htakeda/

[63] Clustering-based denoising with locally learned dictionaries (k-lld). [Online]. Available: http://users.soe.ucsc.edu/~priyam/K-LLD/

[64] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *Image Processing, IEEE Transactions on*, vol. 17, no. 1, pp. 53–69, 2008.

[65] Michael elad. [Online]. Available: http://www.cs.technion.ac.il/~elad/software/

[66] An implementation and detailed analysis of the k-svd image denoising algorithm. [Online]. Available: http://www.ipol.im/pub/art/2012/llm-ksvd/

[67] Julien mairal. [Online]. Available: http://lear.inrialpes.fr/people/mairal/software.php

[68] Image denoising with multi-layer perceptropns. [Online]. Available: http://people.tuebingen.mpg.de/burger/neural_denoising/

[69] M. Lebrun, A. Buades, and J.-M. Morel, "Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm," *Image Processing On Line*, vol. 2013, pp. 1–42, 2013.

[70] Implementation of the "non-local bayes" image denoising algorithm. [Online]. Available: http://www.ipol.im/pub/art/2013/16/

[71] A. Buades, B. Coll, and J.-M. Morel, "Non-Local Means Denoising," *Image Processing On Line*, vol. 2011, 2011.

[72] Non-local means denoising. [Online]. Available: http://www.ipol.im/pub/art/2011/bcm_nlm/

[73] Patch-based locally optimal wiener filtering for image denoising (plow). [Online]. Available: http://users.soe.ucsc.edu/~priyam/PLOW/

[74] Nonlocal sparsity regularized image restoration: a low-rank approach. [Online]. Available: http://www.csee.wvu.edu/~xinl/demo/saist.html

[75] P. Getreuer, "Total Variation Deconvolution using Split Bregman," *Image Processing On Line*, vol. 2012, 2012.

[76] Rudin-osher-fatemi total variation denoising using split berman. [Online]. Available: http://www.ipol.im/pub/art/2012/g-tvd/

[77] Reproducible research in computational science. [Online]. Available: http://www.csee.wvu.edu/~xinl/source.html

[78] F. Porikli, "Constant time o(1) bilateral filtering," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[79] K. Narayan Chaudhury, D. Sage, and M. Unser, "Fast o(1) bilateral filtering using trigonometric range kernels," *IEEE transactions on image processing*, vol. 20, no. 12, pp. 3376–3382, 2011.

[80] F. Rousselle, C. Knaus, and M. Zwicker, "Adaptive rendering with non-local means filtering," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 195, 2012.

Erklärung

gemäss Art. 28 Abs. 2 RSL 05

Name/Vorname: Knaus Claude

Matrikelnummer: 93-915-163 / 272344

Studiengang: Informatik

Bachelor ☐ Master ☐ Dissertation ☒

Titel der Arbeit: Dual-Domain Image Denoising

LeiterIn der Arbeit: Prof. Dr. Matthias Zwicker

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetztes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

Sunnyvale, 14.11.2013

Ort/Datum

C. Knaus

Unterschrift

Biography

Claude Knaus was a Ph.D. student and research assistant at the University of Bern 2009 to 2013. His research interests are rendering and image denoising. Before, he spent 10 years in the industry, including research and development positions at Silicon Graphics Inc. and Object Technology Int. / IBM. He also was involved with the Khronos Group in the specification of OpenGL ES and OpenGL SC. He received his degree in computer science from ETH Zurich in 1998.